[hess??] [gpio??]

Unité Mixte de Recherche 7585 CNRS - LPNHE - IN2P3
Universités Paris VI et Paris VII
4 place Jussieu
Tour 43 rez-de-chaussée
75252 Paris Cedex 05
Tél +033 144 27 63 13
Fax +033 144 27 46 38
http://www-lpnhep.in2p3.fr

PROJET HESS 2

Documentation interne

Nicolas Roche

High Energy Stereoscopic System

29 octobre 2015

PROJET HESS 2

Documentation interne

Nicolas Roche

LPNHE

29 octobre 2015

Résumé

Toute la doc que j'ai pu rassembler et écrire

Mots clés: HESS2, informatique, drivers, interface

Responsables: Mr Pascal VINCENT, Mr François LEGRAND

Annotations:

Table des matières

I Produits **7**

1 Eukréa **11**
1.1 Pré-requis . . . 13
1.1.1 Introduction . . . 14
1.1.2 Serveur TFTP . . . 14
1.1.3 Serveur DHCP . . . 14
1.1.4 Terminal Série . . . 14
1.1.5 Connexion série . . . 15
1.2 Compilation croisée . . . 19
1.2.1 Installation du système cible . . . 20
1.2.2 Modification du système de fichier . . . 23
1.2.3 Modification du noyau linux . . . 24
1.3 Gpio . . . 25
1.3.1 Introduction . . . 26
1.3.2 Localisation des GPIO . . . 26
1.3.3 Test via ssh . . . 27
1.3.4 Programme C . . . 28
1.4 Serveur Tcp . . . 30
1.4.1 Introduction . . . 31
1.4.2 Compilation croisée . . . 31
1.4.3 Modules . . . 31
1.4.4 Tests unitaires de non régression . . . 33

2 Alimentations **35**
2.1 Mib Snmp . . . 38
2.1.1 Installation . . . 39
2.1.2 Requête set . . . 39
2.1.3 Requêtes get . . . 39
2.2 LibSnmp . . . 42
2.2.1 Introduction . . . 43
2.2.2 API SNMP . . . 43
2.2.3 API WIENER . . . 43
2.2.4 Fonctions WIENER . . . 44
2.2.5 Code source . . . 46
2.3 Analyseur grammatical . . . 47
2.3.1 Introduction . . . 48
2.3.2 Grammaire théorique . . . 48
2.3.3 Grammaire implémentée . . . 48
2.4 Intégration . . . 49
2.4.1 Introduction . . . 50
2.4.2 Bibliothèque SNMP . . . 50
2.4.3 Utilitaire WIENER . . . 50

2.4.4 Analyseur Grammatical . . . 50
2.4.5 Test . . . 50

3 Châssis compact-PCI **53**
3.1 Bus cPci . . . 55
3.1.1 Introduction . . . 56
3.1.2 État des lieux (avant les patchs) . . . 56
3.1.3 Mapping depuis le second slot (patch 1) . . . 57
3.1.4 Mapping derrière le second bridge (patch 2) . . . 58
3.1.5 État des lieux (après les patchs) . . . 58
3.1.6 Nouveau code source shl-3.2.3 . . . 59
3.1.7 Carte rio sur le second segment (patch petit nicolas) . . . 61
3.2 Compilation croisée . . . 63
3.2.1 Introduction . . . 64
3.2.2 Test des outils ces sous Debian . . . 64
3.2.3 Cross-compilation sur n1n9 . . . 64
3.2.4 Cross-compilation sur n1n40 . . . 66
3.2.5 Cross-compilation sur Ubuntu . . . 66
3.2.6 Compilation sur cible (carte contrôleur) . . . 67
3.2.7 Debuggage . . . 67
3.3 Boot des cartes contrôleur . . . 69
3.3.1 Introduction . . . 70
3.3.2 Booter depuis le firmware PPCMon . . . 70
3.3.3 Configurations particulières . . . 70
3.3.4 Utilisation des drivers trigers . . . 70
3.3.5 Utilisation de cartes à partir du 2ème segment . . . 71
3.3.6 Plusieurs cartes processeurs . . . 71
3.3.7 Charger les drivers au démarage . . . 71
3.4 Carte ethernet gigabit . . . 73
3.4.1 Introduction . . . 74
3.4.2 Modules . . . 74
3.4.3 Configuration . . . 75
3.4.4 Connectique . . . 77
3.4.5 Tests des débits . . . 77
3.4.6 Test de la fibre optique . . . 78
3.4.7 Test de la configuration finale . . . 78
3.5 dma . . . 80
3.5.1 Introduction . . . 81
3.5.2 User memory transfert . . . 81
3.6 Watch-dog . . . 82
3.6.1 Introduction . . . 83
3.6.2 test . . . 83

4 Contrôleur Sécurité **85**
4.1 Démon Bunny . . . 87
4.1.1 Introduction . . . 88
4.1.2 Répertoires . . . 88
4.1.3 Comportement . . . 88
4.1.4 Remontées des messages d'erreur . . . 88
4.2 Clonage d'un contrôleur Corba . . . 90
4.2.1 Introduction . . . 91
4.2.2 Clonage de l'interface Corba . . . 91
4.2.3 Clonage du serveur Corba . . . 91
4.2.4 Clonage du client non graphique Corba . . . 92

4.2.5 Clonage de l'interface graphique CORBA . . . 93
4.3 Contrôleur CORBA . . . 94
4.3.1 Introduction . . . 95
4.3.2 Architecture du controleur Corba . . . 95
4.3.3 Implémentation du controleur Corba . . . 99

5 Slow Contrôle **101**
5.1 Automates des démons . . . 101
5.1.1 Automate générique . . . 102
5.1.2 Bunny . . . 103
5.1.3 Big . . . 105
5.1.4 Zora *(à faire)* . . . 106
5.1.5 Emilie *(à faire)* . . . 107
5.1.6 Automate des hautes tensions . . . 107
5.1.7 Automate des triggers . . . 108
5.1.8 Automate du capot . . . 108
5.1.9 Todo list . . . 108
5.2 Star Contrôleur . . . 111
5.2.1 Introduction . . . 112
5.2.2 Code HESS1 . . . 112
5.2.3 Extinction préventive . . . 113
5.2.4 Voltage trop élevé . . . 114
5.2.5 Ré-allumage des pixels . . . 114

6 Accéder aux cartes processeurs **115**
6.0.6 Introduction . . . 116
6.0.7 Connexion aux cartes controleurs . . . 116

7 Switchs embarqués **117**
7.0.8 Introduction . . . 118
7.0.9 Connexion aux switchs . . . 118

II Reprise de l'existant **119**

1 Savoir Faire **121**
1.1 Corba . . . 124
1.1.1 Introduction . . . 125
1.1.2 Modèle client/serveur générique . . . 125
1.1.3 Interfaces Dash : :Controller . . . 125
1.1.4 Utilisation de Dash . . . 126
1.2 Interface de pilotage . . . 128
1.2.1 Introduction . . . 129
1.2.2 Interface IDL . . . 129
1.2.3 Contrôleur . . . 129
1.2.4 Client pour les tests . . . 130
1.2.5 Makefile . . . 131
1.2.6 Graphisme . . . 131
1.3 Socket . . . 135
1.3.1 Introduction . . . 136
1.3.2 Client . . . 136
1.3.3 Serveur . . . 137
1.4 Programmes de contrôle . . . 138
1.4.1 Introduction . . . 139

1.4.2 Organisation des répertoires . . . 139
1.4.3 Compilation . . . 139
1.4.4 Fonctionnement des serveurs . . . 140
1.4.5 Grammaire des controleurs . . . 144
1.5 Drivers . . . 146
1.5.1 Introduction . . . 147

2 Outils **149**
2.1 CVS . . . 154
2.1.1 Introduction . . . 155
2.1.2 Commandes . . . 155
2.1.3 Editing administrative files . . . 156
2.1.4 FAQ . . . 157
2.2 TFTP . . . 158
2.2.1 Introduction . . . 159
2.2.2 n1n9 : serveur de boot . . . 159
2.2.3 Carte cliente . . . 159
2.3 NFS . . . 160
2.3.1 Introduction . . . 161
2.3.2 n1n3 : serveur *home* . . . 161
2.3.3 n1n9 : serveur des partitions racines . . . 161
2.3.4 Import des homes sur n1n9 . . . 162
2.3.5 Import des homes sur les cartes processeur . . . 162
2.3.6 Import des partitions racines . . . 163
2.4 DHCP . . . 164
2.4.1 Introduction . . . 165
2.4.2 Configuration du serveur sur n1n9 . . . 165
2.4.3 Configuration des client 'DEBIAN' . . . 165
2.5 makefile . . . 166
2.5.1 Introduction . . . 167
2.5.2 Contrôleurs CORBA . . . 167
2.5.3 Module CVS *SBig* . . . 167
2.5.4 Compilation des librairies embarquées . . . 169
2.6 CORBA . . . 170
2.6.1 Introduction . . . 171
2.6.2 Deamon OMNINAMES . . . 171
2.7 RPM . . . 172
2.7.1 Introduction . . . 173
2.7.2 Commandes en vrac . . . 173
2.7.3 Dépecer un RPM . . . 173
2.7.4 réparer YUM . . . 173
2.8 Pince à sertir . . . 174
2.8.1 Introduction . . . 175
2.8.2 Cable droit . . . 175
2.8.3 Cable croisé . . . 175
2.9 Serveurs de boot . . . 176
2.9.1 Introduction . . . 177
2.9.2 Portable Ubuntu . . . 177
2.9.3 Pizza Boxes . . . 178
2.10 Installation des postes utilisateurs . . . 182
2.10.1 Introduction . . . 183
2.10.2 Préparation de la mise à niveau . . . 183
2.10.3 Installation Fedora 12 . . . 183
2.10.4 Synchronisation temporelle . . . 184

2.10.5 Authentification 184
2.10.6 Installation des softs HESS 185
2.11 Installation des switchs utilisateurs 188
2.11.1 Introduction 189
2.11.2 Netgear GS108 189
2.11.3 Allied AT-800GS/48 190

III Annexes **191**

A Index **195**

Introduction

— Description du projet
Dans le cadre de la collaboration internationale **HESS II**, Le groupe HESS du LPNHE (4 physiciens, 4 électroniciens, 3 informaticiens) est maître d'oeuvre dans la construction de la caméra d'un cinquième télescope de l'expérience qui sera installé en Namibie à la fin de l'année 2009.
Vous trouverez ici, différents articles pour comprendre l'informatique du projet HESS2 :
— La description des systèmes d'information développés.
— Les procédures.
— Les outils disponibles.
— Les connaissances globales sur le projet.
— La documentation.
Merci de prendre le temps de naviguer au sein de cet INTRANET.
— **TODO**
— produit.led : Faire une nouvelle version du controlleur

```
 La carte LED nouvelle version est maintenant sous test. On a enlevé la carte ARM et remplac
La doc est attachée en pdf, regarde à partir du paragraphe 6 p145, c'est relativement compat
merci
Patrick
```

— produit.big : 2 versions à terminer :
— `cvs/SBig_slcReady` à tester
— `cvs/SBig_Wiener` testé
— produit.chassi :
21 slots : mail en cours

```
It cannot ask the RIOC's "resource table", because there is no communication between the
RIOs at that level, except the pci scan itself. All which can be done is to let the RIO in t
system slot assign resources, and let the other RIOs scan the bus and accept the resource
values which are written to the device's config space by the master RIO.

If secondary RIO modifies these resource, this is a problem which I did not expect. In that
I would be nice to have a boot log (or dmesg) of such an event and the following infos from
the two rios:
# cat /proc/bus/xpc/cpci/info
# cat /proc/bus/xpc/cpci/sgi
# cat /proc/bus/xpc/cpci/sgo
# cat /proc/iomem
# cat /proc/ioports
# lspci -nv
you can paste all these info into one file per rio to simplify its aquisition.


I would also like to provide a clean version of the modifications you have done to the
kernel, so could you please tell me which of your "hacks" you are still using/requiring?

To distinguish the slot of the RIOs, it would be possible to use the values shown as "Geo ad
 "Sys slot" of /proc/bus/xpc/cpci/info. (you may have to load the xpc driver with "modprobe
to make sure the file exists)

Best regards,
Rupert
```

— produit.gps :
Prise d'événement externes avant l'initialisation : mail en cours

```
Hello Nicolas,
```

```
I believe that the Model 1088B will not time events when not synchronized to the GPS.
Obviously, the timing would be off and not accurate.

Attached please find a chart that I found on the web that describes the acceptable voltage le
Most of the input/output devices on the MOdel 1088B are HC. For the event input, the device

Best Regards,
Jim
---------------------------------
to: techsupport@arbiter.com

Hi,
I'm trying to record event times using a 1088B Satellite-Controlled Clock.
All is conform to the operation manual when the antenna is fixed and receive at least one sa

The two questions I have are :

1) From 4.8.1 the event/deviation inputs will accept TTL or 5V CMOS. But we only can send 3V
What should be the consequences continuing using 3V event input?

2) Event inputs are ignored when the antenna isn't fixed or doesn't receive at least one sate
How to get event datation (even if erroneous) without the antenna?

Thank you very much for your help.
```

Interface RS232 : centraliser l'intérogation du GPS via Big

— savoirFaire.controleurs :
Automates des serveurs : à mettre en conformité avec l'"Array manager"

— produit.châssis :
RIO sur le deuxième segment : développer un patch du noyau

— outils.serveurs :
Upgrade Fedora 7 : cf cette procédure

```
pas toucher à n1n33 (qui est en redhat 7), mais a priori,
on peut commencer par n1n15...
```

— Compétences informatiques
  — Développement informatique de bas niveau
    — drivers
    — servers C
  — Développement informatique de haut niveau (Corba)
    — controleurs
    — enregistrement et analyse des données
    — array manager
— Personnel du projet

Prénom Nom	métier	savoir faire	contact
Pascal Vincent *	Physique	Dev. bas niveau	vincentp@lpnhe.in2p3.fr
Mathieu de Nauroi	Physique	Dev. haut niveau	01.69.33.55.97 denauroi@in2p3.fr
Jean-Paul Tavernet	Physique	Tests	jean-paul.tavernet@lpnhe.in2p3.fr
François Brun	Physique	Tests	fbrun@lpnhe.in2p3.fr
Julien Bolmont	Physique	Tests	bolmont@lpnhe.in2p3.fr
Patrick Nayman *	Electronique	Conception des cartes	nayman@in2p3.fr
François Toussenel	Electronique	Supervision/Assemblage	francois.toussenel@lpnhe.in2p3.fr
Pascal Corona	Electronique	Tests tirroirs	pascal.corona@lpnhe.in2p3.fr
Jean-marc ...	Electronique	Led	
Jean-François Huppert	Informatique	Dev. bas niveau	jean-francois.huppert@lpnhe.in2p3.fr
Emmanuel Hornero	Informatique	Dev. haut niveau	emmanuel.hornero@lpnhe.in2p3.fr
Richard Randriatoamanana	Informatique	suivi	richard.randria@lpnhe.in2p3.fr
François Legrand	Informatique	suivi	fleg@lpnhe.in2p3.fr
Nicolas Roche	Informatique	Dev.	01.44.27.41.98 nicolasf.roche@gmail.com

* : Responsable de projet.

— Contacts

Prénom Nom	entité	savoir faire	contact
Luisa	CES	Support Produit	luisa@ces.ch
Rupert Eibauer	CES	Support informatique	rupert@ces.ch
Eric Bernard	Eukrea (Armarm9)	Support informatique	ebenard@eukrea.com
Michele Bourgeat	LPTA (Montpellier)	Alims ou Led ?	Michele.Bourgeat@lpta.in2p3.fr
Claude Zurbach	LPTA (Montpellier)	Alims	04.67.14.41.92 Claude.ZURBACH@lpta.in2p3.fr
Jean Luc Panazol	LAP (Annecy-le-Vieux)	Châssis sécurité	04.50.09.16.61 panazol@lapp.in2p3.fr
Thierry le Flour	LAP (Annecy-le-Vieux)	Contrôleur sécurité	04.50.09.17.31 thierry.le-flour@lapp.in2p3.fr
Philippe Venault	CEA (Saclay)	Trigger L2	01.69.08.82.54 philippe.venault@cea.fr

Première partie

Produits

— Serveur embarqué sur carte ARMARM
— Client SNMP interrogeant les alims
— Organisation des cartes sur le chassis
— Serveurs embarqués sur les cartes RIOC 4068
— Controleur CORBA de la carte ctrlSecurity
— Controleur CORBA du trigger L2

Présentation succinte du sytème d'information :

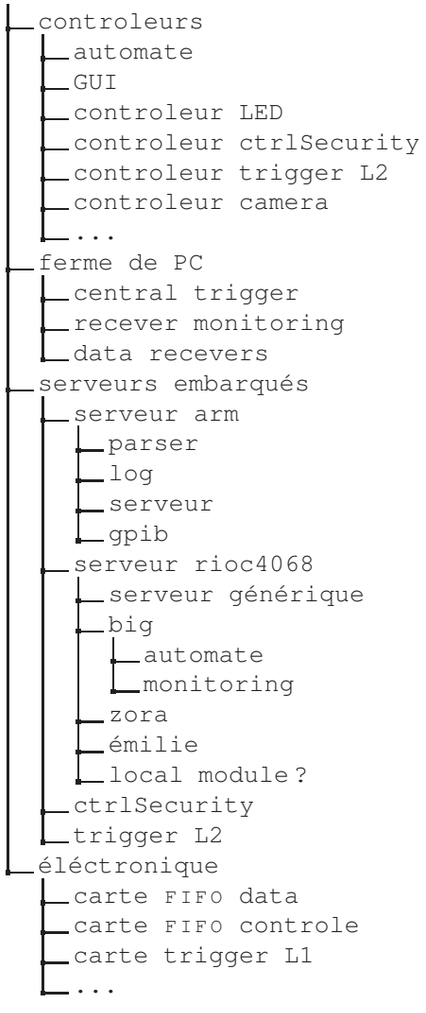


FIGURE 1 – *approche descendante*

Chapitre 1

Eukréa

Il s'agit de développer un système embarqué pour piloter le FPGA responsable du calibrage de la caméra dénommé LED.

Le Kit AT91RM9200 vendu par la société EUKREA fourni une carte de développement disposant d'entrées/sorties afin d'attaquer un 'system on chip' du même nom intégrant à son tour un processeur ARMARM *Thumb* dit ARM920T. Ce Kit fournit par ailleurs un compilateur croisé ainsi qu'un OS LINUX FROM SCRATCH.

Cette documentation expliques les étapes suivies pour de développer un serveur TCP.

— Dans un premier temps nous équipons la mémoire flash du système d'un noyau 2.6 et d'un système de fichier en lecture/écriture (JFFS2). La procédure suivie est calquée sur celle donnée dans la documentation de la carte.
— Suivant toujours la même documentation, nous installerons les outils de cross-compilation.
— Ensuite, nous localiserons et utiliserons les GPIOGPIO afin de programmer un bus de commande de 16 bits. Voici le schéma de la carte dont sont extraites la plupart de ces informations, et le le schéma de plus haut niveau.
— Enfin, nous programmerons une application CLIENT-SERVER qui relayera les requêtes via le réseau.

La carte peut être configuré via DHCP à l'aide de son adresse MAC. Le démon écoute sur toutes les interfaces disponibles et peux être tester via TELNET sur le port 6500 :

```
$ ssh -p 2222 root@arm01
root@arm01's password: eukrea

# tail -f /var/log/led.log
                                                $ telnet arm01 led
                                                Connected to n1n170.in2p3.fr.

led[836]: New connexion from 134.158.155.234:59067

                                                > bye
                                                Connection closed by foreign host.

led[836]: Connexion closed from 134.158.155.234:59067
```

1.1 Pré-requis

1.1.1 Introduction

Grâce à la carte de développement nous pouvons nous interfacer avec le module ARMARM. Cet interfaçage requiert d'installer des logiciels côté client. Nous verrons ici lesquels et comment les configurer.

1.1.2 Serveur TFTP

Le serveurs TFTP dans un premier temps utilisé pour transférer les images du noyau et du système de fichier racine. Ensuite, il reste le meilleur moyen pour uploader de gros fichiers, comme par exemple les binaires à tester.

```
# apt-get install tftpd
# mkdir  /tftpboot
# chown nroche. /tftpboot
# ln -s /tftpboot /src/tftp
```

Le serveur est lancé via INETD. Cf */etc/inetd.conf*.

```
#:BOOT: TFTP service is provided primarily for booting.  Most sites
#       run this only on machines acting as "boot servers."
tftp dgram udp wait nobody /usr/sbin/tcpd /usr/sbin/in.tftpd /srv/tftp
```

1.1.3 Serveur DHCP

En fin de compte on peut se passer du serveur de nom si l'on inscrit en dur une adresse IP sur la carte via par exemple la liaison série.

```
# apt-get install dhcp3-server
```

/etc/dhcp3/dhcpd.conf

```
#option domain-name "example.org";
#option domain-name-servers ns1.example.org, ns2.example.org;

subnet 192.168.1.0 netmask 255.255.255.0 {
         range 192.168.1.2 192.168.1.2;
}
```

1.1.4 Terminal Série

Avant que la liaison ETHERNET soit d'aplomb, on devra utiliser la liaison série pour se connecter à la carte. Voici comment paramétrer la liaison série.

```
# apt-get install gtkterm minicom ckermit lrzsz
# adduser nroche dialoput
```

Les modems sont en mode compressés d'habitude et donc il faut configurer **minicom** en protocole ASCII.

```
$ minicom -s
```

— Configuration du port série

Port série :	/dev/ttyS0
Emplacement du fichier de verrouillage :	/var/lock
Programme d'appel intérieur :	
Programme d'appel extérieur :	
Débit/Parité/Bits :	115200 8N1
Contrôle de flux matériel :	Non
Contrôle de flux logiciel :	Non

— Modem et appel

Chaîne d'initialisation..	(effacé)
Chaîne de remise à zéro..	(effacé)

— Enregistrer config. sous dfl

L'utilisateur normal peut dès lors utiliser minicom :

```
$ minicom
```

Les commandes importantes sont :

CTRL A - Q	quitter
CTRL A - O	configuration
CTRL A - S	envoyer un fichier

1.1.5 Connexion série

Reliez le port série du poste client au port **UART DBG** au moyen d'un câble série croisé (dit femelle - femelle). Positionnez les Jumpers bleu **Jmp9** et **Jmp10** afin de connecter le port série **UART DBG**. Retirez le jumper rouge **Jmp11**.

```
Bienvenue avec minicom 2.3

OPTIONS: I18n
Compilé le Feb 24 2008, 16:35:15.
Port /dev/ttyS0

               Tapez CTRL-A Z pour voir l'aide concernant les touches spéciales


U-Boot 1.2.0 (Mar 21 2008 - 14:15:56)

DRAM:  32 MB
Flash:  8 MB
In:    serial
Out:   serial
Err:   serial
Press SPACE to abort autoboot in 1 seconds
## Booting image at 10040000 ...
   Image Name:   Linux-2.6.22.1
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    1396708 Bytes =  1.3 MB
   Load Address: 20008000
   Entry Point:  20008000
   Verifying Checksum ... OK
OK

Starting kernel ...

Uncompressing Linux.............................................................. \
-->   ................................ done, booting the kernel.
Linux version 2.6.22.1 (eukrea@lpnlp227) (gcc version 4.1.2) #1 PREEMPT \
-->  Fri Mar 21 14:19:32 CET 2008
CPU: ARM920T [41129200] revision 0 (ARMv4T), cr=c0007177
Machine: EUKREA AT91RM9200 SBC
Memory policy: ECC disabled, Data cache writeback
Clocks: CPU 179 MHz, master 59 MHz, main 18.432 MHz
CPU0: D VIVT write-back cache
CPU0: I cache: 16384 bytes, associativity 64, 32 byte lines, 8 sets
```

```
CPU0: D cache: 16384 bytes, associativity 64, 32 byte lines, 8 sets
Built 1 zonelists.  Total pages: 8128
Kernel command line: root=/dev/mtdblock2 rootfstype=jffs2 console=ttyS0,115200 \
-->  mtdparts=physmap-flash.0:256k(u-boot),1408k(kernel),-(rootfs) mem=32M
AT91: 96 gpio irqs in 3 banks
PID hash table entries: 128 (order: 7, 512 bytes)
Console: colour dummy device 80x30
Dentry cache hash table entries: 4096 (order: 2, 16384 bytes)
Inode-cache hash table entries: 2048 (order: 1, 8192 bytes)
Memory: 32MB = 32MB total
Memory: 29480KB available (2612K code, 235K data, 116K init)
Mount-cache hash table entries: 512
CPU: Testing write buffer coherency: ok
NET: Registered protocol family 16
SCSI subsystem initialized
usbcore: registered new interface driver usbfs
usbcore: registered new interface driver hub
usbcore: registered new device driver usb
NET: Registered protocol family 2
IP route cache hash table entries: 1024 (order: 0, 4096 bytes)
TCP established hash table entries: 1024 (order: 1, 8192 bytes)
TCP bind hash table entries: 1024 (order: 0, 4096 bytes)
TCP: Hash tables configured (established 1024 bind 1024)
TCP reno registered
JFFS2 version 2.2. (NAND) ?? 2001-2006 Red Hat, Inc.
io scheduler noop registered
io scheduler deadline registered
io scheduler cfq registered (default)
AT91 Watchdog Timer enabled (5 seconds, nowayout)
atmel_usart.0: ttyS0 at MMIO 0xfefff200 (irq = 1) is a ATMEL_SERIAL
atmel_usart.1: ttyS1 at MMIO 0xfffc0000 (irq = 6) is a ATMEL_SERIAL
atmel_usart.2: ttyS2 at MMIO 0xfffc4000 (irq = 7) is a ATMEL_SERIAL
atmel_usart.3: ttyS3 at MMIO 0xfffc8000 (irq = 8) is a ATMEL_SERIAL
atmel_usart.4: ttyS4 at MMIO 0xfffcc000 (irq = 9) is a ATMEL_SERIAL
RAMDISK driver initialized: 16 RAM disks of 4096K size 1024 blocksize
loop: module loaded
nbd: registered device at major 43
at91_ether: Your bootloader did not configure a MAC address.
eth0: Link down.
eth0: AT91 ethernet at 0xfefbc000 int=24 10-HalfDuplex (00:00:00:00:00:00)
eth0: Micrel KS8721 PHY
physmap platform flash device: 01000000 at 10000000
physmap-flash.0: Found 1 x16 devices at 0x0 in 16-bit bank
 Intel/Sharp Extended Query Table at 0x0031
Using buffer write method
cfi_cmdset_0001: Erase suspend on write enabled
3 cmdlinepart partitions found on MTD device physmap-flash.0
Creating 3 MTD partitions on "physmap-flash.0":
0x00000000-0x00040000 : "u-boot"
0x00040000-0x001a0000 : "kernel"
0x001a0000-0x00800000 : "rootfs"
Generic platform RAM MTD, (c) 2004 Simtec Electronics
mtd-ram mtd-ram.0: registered mtd device
atmel_spi atmel_spi.0: Atmel SPI Controller at 0xfffe0000 (irq 13)
```

```
at91_ohci at91_ohci: AT91 OHCI
at91_ohci at91_ohci: new USB bus registered, assigned bus number 1
at91_ohci at91_ohci: irq 23, io mem 0x00300000
usb usb1: Product: AT91 OHCI
usb usb1: Manufacturer: Linux 2.6.22.1 ohci_hcd
usb usb1: SerialNumber: at91
usb usb1: configuration #1 chosen from 1 choice
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 1 port detected
Initializing USB Mass Storage driver...
      usbcore: registered new interface driver usb-storage
USB Mass Storage support registered.
udc: at91_udc version 3 May 2006
ether gadget: using random self ethernet address
ether gadget: using random host ethernet address
usb0: Ethernet Gadget, version: May Day 2005
usb0: using at91_udc, OUT ep2 IN ep1 STATUS ep4
usb0: MAC 12:c3:b9:85:2d:cf
usb0: HOST MAC 3a:78:c7:96:c2:7e
usb0: RNDIS ready
mice: PS/2 mouse device common for all mice
at91_rtc at91_rtc: rtc core: registered at91_rtc as rtc0
AT91 Real Time Clock driver.
i2c /dev entries driver
at91_i2c at91_i2c: AT91 i2c bus driver.
AT91 MMC: 4 wire bus mode not supported - using 1 wire
TCP cubic registered
NET: Registered protocol family 1
NET: Registered protocol family 17
at91_rtc at91_rtc: setting the system clock to 1998-01-01 00:00:05 (883612805)
VFS: Mounted root (jffs2 filesystem).
Freeing init memory: 116K
eth0: Setting MAC address to 4e:a0:7f:c0:7d:72
var/
var/lib/
var/log/
var/run/
var/tmp/
var/lock/
Starting network...
eth0: Link down.
Starting dropbear sshd: OK
Starting telnet server: OK

CPUAT91 - v2.0 - 2007-06
HESS-2 LED System CPU - LPNHE Paris -
http://www.eukrea.com/ - contact@eukrea.com

 login:
```

Nous sommes dès à présent en possession d'un véritable LINUX FROM SCRATCH. Nul besoin à priori de lui changer quoi que ce soit. Si néanmoins c'était le cas, nous verrons comment réinstaller net OS dans la partie suivante.

Remarque : le système de fichier est monté en lecture/écriture sauf pour les répertoires au contenus volatiles comme */var/* afin de ne pas 'user' la mémoire FLASH.

1.2 Compilation croisée

1.2.1 Installation du système cible

Nous allons dans cette partie récupérer l'environnement de développement fournit avec le Kit de développement :

- Compilation de la chaîne de développement croisée
- Compilation du noyau
- Installation du noyau et du système de fichier sur la cible
- Programmation en C et débogage

```
$ mount /media/cdrom
$ cp /media/cdrom/CPUAT91_20.tar.bz2 .
$ tar -jxf CPUAT91_20.tar.bz2
```

Mise en place des outils de développement

```
# apt-get install gcc flex bison libgmp3-dev libncurses5-dev
```

Attention, bien que les log se plaignent de ne pas avoir LIBEXPAT1-DEV, il ne faut pas l'installer sinon cela génère une erreur concernant ZLIB1G-DEV lors de la compilation de DROPBEAR à l'étape (partie) suivante.

```
$ ./build_tools.sh
```

Ce script construit les principaux utilistaires (crosstool, gdb, mkfs.jffs2, mkcramfs, loader, sx) :

- Si pour chacun d'eux, il ne trouve pas la place d'un marqueur dans le répertoire *stage* alors,
- il copie les sources depuis le répertoire *PKG_CPUAT91*
- dans l'espace de compilation : *build*. **(écrasement)**
- les résultats de la compilation sont visible dans le répertoire *log*
- et les executables seront pour la majeur partie rangés dans le répertoire *tools* et accessoirement *images*

Bootloader et noyau

```
# apt-get install libacl1-dev
```

Remarque, le paquet ZLIB1G-DEV semble être facultatif.

```
$ ./build_distrib.sh
```

De façon trés similaire au script précédent, celui-ci construit les composantes de notre OS :

- Si pour chacune d'elles, il ne trouve pas la place d'un marqueur dans le répertoire *stage* alors,
- il copie les sources depuis le répertoire *PKG_CPUAT91*
- dans l'espace de compilation : *build*. **(écrasement)**
- les résultats de la compilation sont visible dans le répertoire *log*
- et les executables seront pour la majeur partie rangés dans le répertoire *rootfs* et accessoirement *images*.
- efin, le script crée les images compressées du système de fichier. (ce que fait aussi le script *build_image.sh*).

Voici les composantes :

- uboot
  - *images/u-boot_ram.bin* chargé par le script *flash_cpuat91.sh*
  - *tools/gcc-4.1.2-glibc-2.3.6/arm-linux/bin/mkimage*
- linux
- rootfs
- busybox
- gdb
- gdbserver

- strace
- eeprog
- gpiogpio
- dosfs
- devmem2
- cpuat91
- libs (anl BrokenLocale c crypt dl gcc_s m memusage nsl nss_compat nss_dns nss_files nss_hesiod nss_nis nss_nisplus pcprofile pthread resolv rt SegFault stdc++ thread_db util zlib)
- modules

Installation LINUX FROM SCRATCH

La procédure suivante permet de réinstaller notre OS. Elle constera grosso-mod à :
- Charger le programme (`sx-at91 -s /dev/ttyS0 images/loader.bin`)
- Charger le shell utilisé par la suite (`sx-at91 -s /dev/ttyS0 images/u-boot_ram.bin`)
- Charger le bootloader, le noyau et l'image du système de fichier.
  - */tftpboot/u-boot_cpuat91.bin*
  - */tftpboot/uImage_cpuat91*
  - */tftpboot/rootfs_cpuat91.jffs2*

Relier le port série du poste client au port **UART DBG** au moyen d'un câble série croisé (dit femelle - femelle). Positionner les Jumpers bleu **Jmp9** et **Jmp10** afin de connecter le port série **UART DBG**. **Connecter** le jumper rouge **Jmp11**.

Pensez à appuyer sur le bouton reset (le bouton jaune 'RST').

```
$ cp images/u-boot_cpuat91.bin images/uImage_cpuat91 images/rootfs_cpuat91.jffs2 /tftpboot
$ ./flash_cpuat91.sh /dev/ttyS0
```

Une fois U-BOOT flashé, on reconnecte le terminal série. **Remarque :** l'adresse mac entrée ci-dessous est facultative.

```
$ minicom

CPUAT91=> help
CPUAT91=> ping 192.168.1.1
Micrel KS8721 PHY detected : 100BT FD
host 192.168.1.1 is alive
CPUAT91=> ethaddr 0a:03:4a:c6:f3:a2
...

setenv ipaddr 192.168.1.2;
setenv serverip 192.168.1.1;
saveenv

tftp 20000000 u-boot_cpuat91.bin;
protect off 10000000 1001FFFF;
erase 10000000 1001FFFF;
cp.b 20000000 10000000 ${filesize};
protect on 10000000 1001FFFF

tftp 20000000 uImage_cpuat91;
protect off 10040000 1019FFFF;
erase 10040000 1019FFFF;
cp.b 20000000 10040000 ${filesize}

tftp 20000000 rootfs_cpuat91.jffs2;
```

```
protect off 101A0000 107FFFFF;
erase 101A0000 107FFFFF;
cp.b 20000000 101A0000 ${filesize}

setenv bootargs root=/dev/mtdblock2 rootfstype=jffs2 console=ttyS0,115200 \
mtdparts=physmap-flash.0:256k(u-boot),1408k(kernel),-(rootfs) mem=32M

setenv bootcmd bootm 10040000;
setenv bootdelay 1;
saveenv
```

Résumé : (pour copier/coller) :

- setenv ipaddr 192.168.1.2 ; setenv serverip 192.168.1.1 ; saveenv
- tftp 20000000 u-boot_cpuat91.bin; protect off 10000000 1001FFFF ; erase 10000000 1001FFFF ; cp.b 20000000 10000000 ${filesize} ; protect on 10000000 1001FFFF
- tftp 20000000 uImage_cpuat91 ; protect off 10040000 1019FFFF ; erase 10040000 1019FFFF ; cp.b 20000000 10040000 ${filesize}
- tftp 20000000 rootfs_cpuat91.jffs2 ; protect off 101A0000 107FFFFF ; erase 101A0000 107FFFFF ; cp.b 20000000 101A0000 ${filesize}
- setenv bootargs root=/dev/mtdblock2 rootfstype=jffs2 console=ttyS0,115200 mtdparts=physmap-flash.0 :256k(u-boot),1408k(kernel),-(rootfs) mem=32M ; setenv bootcmd bootm 10040000 ; setenv bootdelay 1 ; saveenv

cf (on sait jamais) `#define CONFIG_EXTRA_ENV_SETTINGS` dans *build/u-boot-1.2.0/include/configs/cpuat91.h*

Pour tester, retirer le jumper rouge **Jmp11** et générez un reset en pressant le bouton.

Via le port série :

```
eukrea  login: root
Password: eukrea
login[864]: root login on 'ttyS0'
```

Via SSH :

```
$ ssh root@192.168.1.2 -p 2222
root@192.168.1.2's password: eukrea
#
```

Développement C

Sur le PC hôte :

```
./setup_env.sh
Configuration de l'environnement pour la compilation croisée
Vous êtes à présent dans l'environnement de cross-compilation
Tapper CTRL+D pour restaurer l'environnement initial2

$ cd exemples/
$ make
```

Sur la carte :

```
# cd /tmp
# tftp -g -r test 192.168.1.1
# chmod +x test
# ./test
Hello World !
i = 15 - j = 12

# gdbserver 192.168.1.5:2345 ./test
```

A nouveau sur le PC hôte :

```
$ arm-linux-gdb test_g

(gdb) target remote 192.168.1.2:2345
...
```

Afin de faire communiquer la carte nous nous intéresserons, dans la partie suivante.

1.2.2 Modification du système de fichier

Il s'agit d'une sous partie de l'algorithme exposé ci-dessus. Le script *rebuild_distrib.sh* va construire pas à pas le système de fichier. Il sera :

- Décompressé depuis *PKG_CPUAT91/rootfstree-cpuat91.tar.bz2* **(écrasement)**
- Enrichi des composantes *busybox devmem2 dosfs dropbear eeprog gpiogpio gdbserver libs rootfs strace zlib modules et cpuat91*.
- Compréssé en 2 images : *rootfs_cpuat91.cramfs* et *rootfs_cpuat91.jffs2* la première aboutissant à un système de fichier en lecture seule tandis que la seconde permet de modifier directement la mémoire flash bien que toutefois les logs soient enregistrées en RAM afin de l'économiser.

Modifier l'adresse mac temporaire (non requis) :

```
$ ./build/u-boot-1.2.0/tools/gen_eth_addr
1a:94:65:70:e0:bf
```

Nous allons à présent modifier l'archive contenant le système de fichier utilisé.

```
$ cd PKG_CPUAT91
$ mkdir tmp
$ cd tmp
$ tar -xjf ../rootfstree-cpuat91.tar.bz2
```

fichier *etc/network/interface* :

```
# Configure Loopback
auto lo eth0
iface lo inet loopback

iface eth0 inet static
       address 192.168.1.2
       netmask 255.255.255.0
       gateway 192.168.1.1
```

fichier *etc/hostname* :

```
eukrea
```

fichier *etc/hosts* :

```
127.0.0.1       localhost
```

fichier *etc/issue* :

```
 -----------------
( Agate the bouze )
 -----------------
        o   ^__^
         o  (xx)\\_______
            (__)\\       )\\/\\
             U  ||----w |
                ||     ||
```

Il faut refermer l'archive contenant le système de fichier puis reconstruire l'image.

```
$ tar -cjf ../rootfstree-cpuat91.tar.bz2 .
$ cd ../..
$ ./rebuild_distrib.sh
$ cp images/* /tftpboot/
```

Le script *rebuild_distrib.sh* invoque le script *build_distrib.sh* décrit ci-dessus aprés avoir effacé les stages *busybox devmem2 dosfs dropbear eeprog gpiogpio gdbserver libs rootfs strace zlib modules et cpuat91*.

A tester : configuration du réseau via DHCP.

1.2.3 Modification du noyau linux

Il faut suivre manuellement les instruction du script *build_distrib.sh*. En effet, le script n'aura aucun effet si les fichiers 'stages' ne sont pas détruits. En revanche s'il le sont, les sources seront écrasées avec toute vos modifications.

Certaines GPIOGPIO ne sont pas utilisables tel que (y compris en retirant les jumper). Il faut libérer les GPIOGPIO relatives aux UARTS 0, 1 et 3 dans le fichier *build/linux-2.6.22.1/arch/arm/mach-at91/board-cpuat91.c* :

```
/*
 * Serial port configuration.
 *    0 .. 3 = USART0 .. USART3
 *    4      = DBGUG
 */
static struct at91_uart_config __initdata cpuat91_uart_config = {
.console_tty = 0,/* ttyS0 */
.nr_tty = 5,
.tty_map = { 4, 2} /* ttyS0, ..., ttyS4 */
};
```

Recompilation et installation :

```
$ cd build/linux-2.6.22.1
$ make menuconfig

$ vi arch/arm/mach-at91/board-cpuat91.c
$ vi arch/arm/mach-at91/at91rm9200_devices.c

$ make uImage modules
$ make INSTALL_MOD_PATH=../../rootfs/ modules_install
$ cp arch/arm/boot/uImage ../../images/uImage_cpuat91
$ cd ../..
$ ./build_images.sh
$ cp images/* /tftpboot/
```

Réinstallez linux comme décrit deux paragraphes plus haut.

1.3 Gpio

1.3.1 Introduction

La carte de développement met à notre dispositon un ensemble de bits dits GPIOGPIO rendu accessibles au câblage via deux pseudo bus parallèles. Ces bits sont analogues à ceux du port parallèle. J'ai étudié les GPIOGPIO dans le contexte d'une donnée inscrite en sortie (pas en entrée). Ces bits sont alimentés à 3,3 volts lorsqu'ils sont positionnés à 1. Nous verrons dans cette partie, comment commander ces bits par bloc de 16.

1.3.2 Localisation des GPIO

Les pseudo ports parallèle **J10** et **J7** nous donnent accès à :
— La masse, 5v et 3,3V
— Certaines broche du module AT91RM9200 hors GPIO
— Des GPIO utilisées par le module
— Des GPIO déconnectables du module via des jumpers
— Des GPIO libres

Affectation *théorique* des GPIO

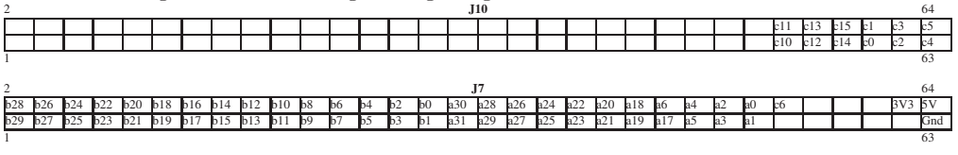
La localisation des GPIOGPIO est donnée par ces 2 schémas :
— le schéma de la carte
— le schéma de plus haut niveau
GPIO directement à des composants :
— **I2C**
  — a25, a26
— **USB**
  — c14, c15
— **SDCARD/MMC**
  — a27, a28, a29
  — b3, b4, b5
  — c2
— nWAIT
  — c6
GPIO à des composants via jumpers :
— **Led**
  — c0
  — Jmp23
— **UART 0**
  — a17, a18, a20, a21
  — Jmp14, à Jmp17
— **UART 3**
  — a5, a6
  — b0, b1
  — JM18 à Jmp21
— **UART Debug & 2**
  — a22, a23, a30, a31
  — Jmp9, Jmp10, Jmp12 et Jmp13
— **UART 1**
  — b18, b19, b20, b21, b23, b24, b25, b26
  — Jmp1 à Jmp8
— **Reset** (difficilement exploitable puisque réservé au flashage)
  — a31
  — Jmp11

En pratique

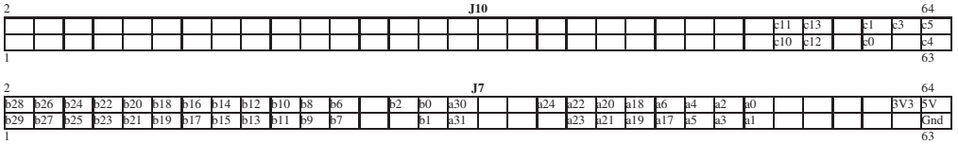
- Famille C (c[0-6] et c[10-15]) répartie sur les 12 bits de poids forts des 2 bus.
  - c0 commande la led (non utilisable)
  - c6 utilisé par *nWait* (utilisation déconseillé) ?
  - c2 utilisé par SDCARD/SSC ?
  - c14, c15 utilisés par USB ?
  - c1, c3, c4, c5, c[10-13] : 8 non affectés ?
- Famille B (b[0-29]) répartie sur les 30 bits de poids faibles du bus J7.
  - b0, b1 utilisés par UART 3 (utilisables)
  - b[3-5], utilisés par SDCARD/SSC (utilisation non réussie : dé-sélectionner **driver/MMC** lors de la compile du noyau)
  - b[12-19], utilisés par *MAC+PHY Ethernet* (cf *build/linux-2.6.22.1/arch/arm/mach-at91/at91rm9200_devices.c*)
  - b[18-21] et b[23-26] utilisé par UART 1 (utilisable sauf b21 : 2v pour 1 logique)
  - b2, b[6-11], b22 et b[27-29] : 8 bits non affectés ?
- Famille A (a[0-6] et a[17-31]) répartie sur les 22 derniers bits au milieu du bus J7.
  - a5 et a6 utilisés par UART 3 (utilisables)
  - a17, a18, a20 et a21 utilisés par UART 0 (utilisables)
  - a22, a23 utilisés par UART 2 (utilisables)
  - a25, a26 utilisés par I2C (utilisables : déselection du driver à la compil du noyau ?)
  - a[27-29] utilisés par SDCARD/SSC (utilisables ? !)
  - a30 et a31 utilisés par UART DEBUG (non utilisables : font planter la carte)
  - a[0-4], a19, a24 : 7 bits non affectés (utilisables directement)

FIGURE 1.1 – Disposition des GPIO sur les deux pseudo ports parallèles

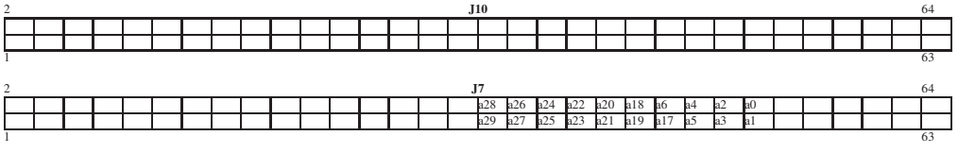
Voici les GPIO présentes sur les 2 pseudo ports parallèle **J10** et **J7**.



Les GPIO testées libres avec la distribution de base (à réaliser) :



Les GPIO testées comme rendues libres (à compléter) :



1.3.3 Test via ssh

On peut éteindre puis rallumer la led du PC Client par exemple :

```
$ ssh-keygen -t dsa
$ cat ~/.ssh/id_dsa.pub | ssh root@192.168.1.2 -p 2222 'cat > ~/.ssh/authorized_keys'
```

```
$ ssh root@192.168.1.2 -p 2222 gpio -PC0
$ ssh root@192.168.1.2 -p 2222 gpio +PC0
```

Ce petit programme d'exemple permet de positionner les d'un coup toutes les GPIO modifiables. Attention toucher à **a30** ou à **a31** freeze la carte. Les bits doivent être testé au voltmètre ou à l'osciloscope.

```
#!/bin/sh

echo "Ce script bascule l'ensemble des GPIO A, B et C."
Usage()
{
    echo -e "Usage: \n\t$1 {+|-|?}"
    exit 1
}

[ $# -eq 1 ] || Usage $0
[ $1 == "+" ] || [ $1 == "-" ] || [ $1 == "?" ] || Usage $0

listA=" a0  a1  a2  a3  a4  a5  a6              \
                                a17 a18 a19 \
        a20 a21 a22 a23 a24 a25 a26 a27 a28 a29 "
listB=" b0  b1  b2  b3  b4  b5  b6  b7  b8  b9  \
        b10 b11 b12 b13 b14 b15 b16 b17 b18 b19 \
        b20 b21 b22 b23 b24 b25 b26 b27 b28 b29 "
listC=" c0  c1  c2  c3  c4  c5  c6              \
        c10 c11 c12 c13 c14 c15                 "


# $1 : bus name
# $2 : list
LoopOnList()
{
        echo -ne "$1:\t"
        for io in $2
        do
                [ $1 != "?" ] && gpio ${1}P${io}
                gpio ?P${io}
                echo -n "$? "
        done
}

[ $1 == "?" ] || echo "mise a $1 ..."
LoopOnList "A"  "$listA"
LoopOnList "B"  "$listC"
LoopOnList "C"  "$listC"
echo
```

1.3.4 Programme C

On peut donc utiliser les GPIOGPIO **a[0-6]** et **a[17-25]** afin d'avoir 16 bits consécutifs.
Les procédures suivantes permettent d'utiliser 16 bits avec :

— pour bit de poids fort (utilisé comme bit *strobe*) la GPIOGPIO **a25**.
— pour bit de poids faible la GPIOGPIO **a0**.

```
#include <memory.h>
#include <sys/mman.h>
#include <sys/ioctl.h>
#include <fcntl.h>
...

int mapPage(int *fd, void **map_base)
{
  int rc = 1;

  if((*fd = open("/dev/mem", O_RDWR | O_SYNC)) == -1){
    exit(-1);
  }

  /* Map one page */
  *map_base = mmap(0, MAP_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, *fd, AT91_SYS & ~MAP_MASK);
  if(*map_base == (void *) -1){
    close(fd);
    exit(-1);
  }

  return rc;
}

int unmapPage(int *fd, void **map_base)
{
  int rc = 1;

  /* Unmap previously mapped page */
  if(munmap(*map_base, MAP_SIZE) == -1){
    close(fd);
    exit(-1);
  }

  close(*fd);
  *fd = 0;
  *map_base = (void *) 0;
  return rc;
}

/* return GPIO from a0-a6 and a17-a25 */
unsigned int getLedGpio()
```

```
{
  int fd, pin,i;
  void *map_base;
  unsigned long readval, writeval, lowVal, hightVal;
  unsigned long lowMask = 0x0000007F;
  unsigned long hightMask = 0x03FE0000;
  unsigned int  combinedVal;

  int dest_addr = PIOA_OFFSET;
  int id = PIOA_ID;

  mapPage(&fd, &map_base);

  /* First, enable PIO clock */
  writeval = 1 << id;
  *((unsigned long*)(map_base + ((PMC_OFFSET + PMC_PCER) & MAP_MASK))) = writeval;
  /* Then read port value */
  readval = *((unsigned long*)(map_base + ((dest_addr + PIO_PDSR) & MAP_MASK)));

  lowVal = readval & lowMask;
  hightVal = (readval & hightMask) >> 10;
  combinedVal = hightVal | lowVal;

  unmapPage(&fd, &map_base);
  return combinedVal;
}

/* set GPIO from a0-a6 and a17-a25 */
int setLedGpio(unsigned int val)
{
  int rc = 1, fd, pin;
  void *map_base;
  unsigned long writeval, lowVal, hightVal, combinedVal;
  unsigned int mask = 0xFFFF;
  unsigned int lowMask = 0x007F;
  unsigned int hightMask = 0xFF80;

  int dest_addr = PIOA_OFFSET;
  int id = PIOA_ID;

  if ((val | mask) != mask) rc = -1;
  lowVal = val & lowMask;
  hightVal = ((val & hightMask) >> 7) << 17;
  combinedVal = lowVal | hightVal;

  mapPage(&fd, &map_base);

  /* set GPIO from a0 to a6 */
  for (pin=0; pin<=6; ++pin)
    {
      /* Setup PIO registers */
      writeval = 1<< pin;
      *((unsigned long*)(map_base + ((dest_addr + PIO_PER) & MAP_MASK))) = writeval;
      *((unsigned long*)(map_base + ((dest_addr + PIO_OER) & MAP_MASK))) = writeval;

      if ((combinedVal >> pin) & 0x00000001 == 1)
{
  /* set */
  *((unsigned long*)(map_base + ((dest_addr + PIO_SODR) & MAP_MASK))) = writeval;
}
      else
{
  /* unset */
  *((unsigned long*)(map_base + ((dest_addr + PIO_CODR) & MAP_MASK))) = writeval;
}
    }
 /* set GPIO from a17 to a25 */
  for (pin=17; pin<=25; ++pin)
    {
      /* Setup PIO registers */
      writeval = 1<< pin;
      *((unsigned long*)(map_base + ((dest_addr + PIO_PER) & MAP_MASK))) = writeval;
      *((unsigned long*)(map_base + ((dest_addr + PIO_OER) & MAP_MASK))) = writeval;

      if ((combinedVal >> pin) & 0x00000001 == 1)
{
  /* set */
  *((unsigned long*)(map_base + ((dest_addr + PIO_SODR) & MAP_MASK))) = writeval;
}
      else
{
  /* unset */
  *((unsigned long*)(map_base + ((dest_addr + PIO_CODR) & MAP_MASK))) = writeval;
}
    }

  unmapPage(&fd, &map_base);
  return rc;
}
...
int strobeIt(unsigned int paio)
{
  int rc = 1;
  rc &= setLedGpio(paio & 0x7FFF);
  logEmit(DefaultLog, "LED GPIO: 0x%4x\n", getLedGpio());

  rc &= setLedGpio(paio | 0x8000);
  logEmit(DefaultLog, "LED GPIO: 0x%4x\n", getLedGpio());

  rc &= setLedGpio(paio & 0x7FFF);
  logEmit(DefaultLog, "LED GPIO: 0x%4x\n", getLedGpio());
  return rc;
}
es
```

1.4 Serveur Tcp

1.4.1 Introduction

Il s'agit de développer un serveur suivant des méthodes de développement éprouvée. Aussi nous utiliserons une approche modulaire permettant une construction pas à pas et ainsi la mise en place de tests unitaires de non régression.

1.4.2 Compilation croisée

Avant de tester sur la cible, on peut compiler puis tester en local, comme nous le voyons dans le préambule de ce *Makefile*.

```
targetSystem = $(shell uname -s)
targetProcessor = $(shell echo $(PATH) | grep -q 'arm-linux' && echo "arm" || echo "i386")

ifneq ($(targetSystem),Linux)
$(error unknown target system '$(targetSystem)')
endif

ifdef DEBUG
optimizingCFLAGS = -g -O0
optimizingLDFLAGS = -g
else
optimizingCFLAGS = -O2
optimizingLDFLAGS = -Xlinker -s
endif

commonCFLAGS = \
$(optimizingCFLAGS) \
-I include

CFLAG = -c
utCFLAGS = -D_utMAIN
LDFLAGS = $(optimizingLDFLAGS)

ifeq ($(targetProcessor),i386)
CC = gcc
CFLAGS = $(commonCFLAGS)
else
ifeq ($(targetProcessor),arm)
CC       =  arm-linux-gcc
C++ =  arm-linux-g++
CFLAGS   =  \
$(commonCFLAGS) \
-mcpu=arm920t \
-mtune=arm920t \
-msoft-float
else
$(error unknown target processor '$(targetProcessor)')
endif
endif
```

1.4.3 Modules

L'architecture se découpe en module. Nous les présentons dans l'ordre des dépendances. Chaque module est compilé sous forme d'objet mais également sous forme d'un exécutable permettant d'effectuer ses

tests.

IO

Ce module regroupe les primitives d'entrée/sortie. Il devrait à terme présenter une abstraction des méthodes d'accès aux fichiers que ce soit via les **handler** ou les **descripteurs**. En effet, alors que l'accès aux fichiers normaux se fait via des fonctions de haut niveaux, les sockets comme d'autres fichiers spéciaux ne sont accessibles que uniquement via les fonction de bas niveau car ces accès ne sont pas buffurisés. Pour nos besoins, seule l'écriture est implémenté avec les descripteurs de fichiers.

Log

Ce module regroupe les primitives d'écriture des messages d'erreurs. Étant donné que notre LINUX FROM SCRATCH ne possède pas de démon SYSLOG, nous limitons ici nos besoins à l'écriture dans un simple fichier de LOG.

Gpio

Nous avons décrit ce module dans la partie précédente.

Lexer

Bien que notre protocole soit très simple, nous utilisons le Lexer FLEX pour identifier les symboles terminaux de notre grammaire.

Parser

Encore une fois, bien que notre protocole soit très simple, nous utilisons le Parser BISON pour programmer l'automate correspondant à la grammaire de notre protocole.

Serveur Tcp

Ce module écoute sur un port donné **à définir dans *etc/services***. Pour raison de commodité lors des changement d'adresses, le serveur écoute sur toutes les adresses IP. Ce module se met personnelement en relation avec les sockects clientes afin de garantir un accès exclusif au serveur.

Daemon

Ce petit bout de code à pour but de désolidariser le processus d'une session utilisateur et d'une console. Ainsi, le programme pourra tourner en tâche de fond en étant géré par les scripts d'initialisation de type SYSTEM-V. Voir le fichier */etc/init.d/S99led* :

```
#!/bin/sh

start() {
        /opt/led
}

stop() {
        killall led
}

restart() {
        stop
        start
```

```
}

case "$1" in
  start)
        start
        ;;
  stop)
        stop
        ;;
  restart|reload)
        restart
        ;;
  *)
        echo $"Usage: $0 {start|stop|restart}"
        exit 1
esac

exit $?
```

1.4.4 Tests unitaires de non régression

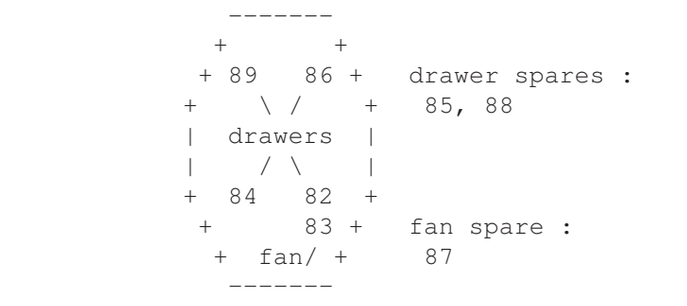
Il s'agit de tester chaque module indépendamment des autres ou alors dans un ordre permettant de lever les ambiguïtés sur la responsabilité de chacun des module sur les erreurs détectées. Le résultat des test est comparé au résultat attendu afin de s'assurer de ne pas avoir de régression.

```
$ make obj/io.o test/utio

$ test/utio -h
The usage for test/utio is:
        test/utio { -h | [ -i input ] [ -o output ] }   where:
                -i   : input stream
                -o   : output stream
                -h   : help

$ test/utio
text saisi recopié ci-dessous
text saisi recopié ci-dessous
^D
```

Chapitre 2

Alimentations

La caméra contient 5 alimentations dont une haute tension. (cf lien vers le bus de la caméra)
Les alimentations sont controlable via le protocole SNMP implémenté par la librairie *libnetsnmp*.



— Interface web des alims (adresse par défaut) :

```
pizza41# ifdown eth0
pizza41# ifconfig eth0 192.168.91.1
pizza41# route add -net 192.168.91.80 netmask 255.255.255.255 dev eth0

lpnp90# /sbin/route add -host 192.168.91.80 gw 192.168.1.41
(marche pas sur mon poste debian !?)
```

— Changer l'adresse IP :

Select-Mode haut/bas	Aller à la position précédent TCPIP
Select-Mode haut + *Power* haut	maintenir les deux boutons 5 secondes
Select-Mode haut/bas	Aller à la position adresse TCPIP
Power haut	pour changer d'octer et valiser
Select-Mode haut/bas	pour changer la valeur de l'octet
Power bas	pour sortir du menu

— Il faut implémenter (*italique = ce qui n'est pas fait*) :
  — put état (on/off)
  — *get état (on/off)*
  — *get tension terminale par alims*
  — *get ampérage par alims*
  — *get température globale*
  — *get température par alims*

  Pour la mise sous tension, il faut :
  — mise sous tension
  — attendre que la tension se stabilise
  — *décider si la tension est accéptable*
  — comparer la tension à sa constante max
  — retourner un code d'erreur

  Rq : les routines sont extraites des fichier suivant :
  — */SBig/Driver/Wiener/snmp/getwiener.c*
  — */SBig/Driver/Wiener/snmp/setwiener.c*
  — la valeur -128 signifie : 'débranché'
  — la valeur 127 signifie : 'non implémenté'

  Par exemple :

```
./getwiener 192.168.1.81
```

  rq : l'installation de la librairie est détaillée dans la rubrique châssis/cross-compile :

```
#apt-get install libsnmp-base libsnmp-dev
```

  Critères retenus pour le contrôle des alims :

```
19947.1.3.2.1.5.1  outputMeasurementSenseVoltage.U0 = Opaque: Float: 5.010000 V
19947.1.3.2.1.16.1 outputSupervisionMinSenseVoltage.U0 = Opaque: Float: 4.750000 V
19947.1.3.2.1.17.1 outputSupervisionMinSenseVoltage.U6 = Opaque: Float: 28.500000 V
```

```
19947.1.3.2.1.7.1 outputMeasurementCurrent.U0 = Opaque: Float: 0.040000 A
19947.1.3.2.1.19.1 outputSupervisionMaxCurrent.U0 = Opaque: Float: 30.000000 A
```

2.1 Mib Snmp

2.1.1 Installation

```
#apt-get snmp

$ snmpwalk -v 2c -c public -On 192.168.1.81 .
...
.1.3.6.1.2.1.1.7.0 = INTEGER: 79
.1.3.6.1.4.1.19947.1.1.1.0 = INTEGER: 1
...
```

Afin d'obtenir le nom des noeuds, il faut préalablement ajouter la MIB spécifique *WIENER-CRATE-MIB.txt* côté client.

```
$ snmpwalk -v 2c -c public -Of 192.168.1.81 .
...
.iso.org.dod.internet.mgmt.mib-2.system.sysServices.0 = INTEGER: 79
.iso.org.dod.internet.private.enterprises.19947.1.1.1.0 = INTEGER: 1
...

$ mkdir -p ~/.snmp/mibs
$ cp WIENER-CRATE-MIB.txt ~/.snmp/mibs
$ export MIBS=+WIENER-CRATE-MIB
$ snmpwalk -v 2c -c public -Of -m +WIENER-CRATE-MIB 192.168.1.81 .
...
.iso.org.dod.internet.mgmt.mib-2.system.sysServices.0 = INTEGER: 79
.iso.org.dod.internet.private.enterprises.wiener.crate.system.sysMainSwitch.0 = INTEGER: ON(1)
...
```

The SNMP community names for different views. The rights of the different communities are :
- public : no write access
- private : can switch power on/off, generate system reset
- admin : can change supervision levels
- guru : can change output voltage & current (this may destroy hardware if done wrong !)

2.1.2 Requête set

Cette commande alume l'alimentation *192.168.1.86*.

```
$ snmpset -v 2c -c admin -Of 192.168.1.86 .1.3.6.1.4.1.19947.1.1.1.0 i 1
.iso.org.dod.internet.private.enterprises.19947.1.1.1.0 = INTEGER: 1
```

2.1.3 Requêtes get

Informations :
- get état (on/off)

```
sysStatus OBJECT-TYPE
    SYNTAX  BITS {
        mainOn (0) ,
        mainInhibit (1) ,
        localControlOnly (2) ,
        inputFailure (3) ,
        outputFailure (4) ,
        fantrayFailure (5) ,
        sensorFailure (6),
        VmeSysfail (7),
        plugAndPlayIncompatible (8)
    }
    MAX-ACCESS  read-only
    STATUS  current
    DESCRIPTION
            "A bit string which shows the status (health) of the complete crate.
             If a bit is set (1), the explanation is satisfied
               mainOn (0),                    system is switched on, individual outputs
                                              may be controlled by their specific ON/INHIBIT
               mainInhibit(1),    external (hardware-)inhibit of the complete system
               localControlOnly (2),local control only (CANBUS write access denied)
               inputFailure (3),any input failure (e.g. power fail)
```

```
                outputFailure (4),any output failure, details in outputTable
                fantrayFailure (5),fantray failure
          sensorFailure (6),           temperature of the external sensors too high
          VmeSysfail(7),   VME SYSFAIL signal is active
          plugAndPlayIncompatible (8) wrong power supply and rack connected
          "

outputStatus OBJECT-TYPE
    SYNTAX  BITS {
        outputOn (0) ,
        outputInhibit (1) ,
        outputFailureMinSenseVoltage (2),
        outputFailureMaxSenseVoltage (3),
        outputFailureMaxTerminalVoltage (4),
        outputFailureMaxCurrent (5),
        outputFailureMaxTemperature (6),
        outputFailureMaxPower (7),
        -- reserved
        outputFailureTimeout (9),
    outputCurrentLimited (10),
    outputRampUp (11),
    outputRampDown (12),
    }
    MAX-ACCESS  read-only
    STATUS  current
    DESCRIPTION
            "A bit string which shows the status (health and operating conditions) of one output channel.
             If a bit is set (1), the explanation is satisfied:
                outputOn (0),output channel is on
                outputInhibit(1),external (hardware-)inhibit of the output channel
         outputFailureMinSenseVoltage (2), Supervision limit hurt: Sense voltage is too low
         outputFailureMaxSenseVoltage (3), Supervision limit hurt: Sense voltage is too high
         outputFailureMaxTerminalVoltage (4), Supervision limit hurt: Terminal voltage is too high
         outputFailureMaxCurrent (5),     Supervision limit hurt: Current is too high
         outputFailureMaxTemperature (6), Supervision limit hurt: Heat sink temperature is too high
         outputFailureMaxPower (7),Supervision limit hurt: Output power is too high
                outputFailureTimeout (9),  Communication timeout between output channel and main control
outputCurrentLimited (10),      Current limiting is active (constant current mode)
    outputRampUp (11),      Output voltage is increasing (e.g. after switch on)
    outputRampDown (12),    Output voltage is decreasing (e.g. after switch off)"

.1.3.6.1.4.1.19947.1.3.2.1.9.1 = INTEGER: ON(1)
.1.3.6.1.4.1.19947.1.3.2.1.9.2 = INTEGER: ON(1)
.1.3.6.1.4.1.19947.1.3.2.1.9.4 = INTEGER: ON(1)
.1.3.6.1.4.1.19947.1.3.2.1.9.5 = INTEGER: ON(1)

.1.3.6.1.4.1.19947.1.1.2.0 = BITS: 80 00 mainOn(0)
.1.3.6.1.4.1.19947.1.3.2.1.4.1 = BITS: 80 outputOn(0)
.1.3.6.1.4.1.19947.1.3.2.1.4.2 = BITS: 80 outputOn(0)
.1.3.6.1.4.1.19947.1.3.2.1.4.4 = BITS: 80 outputOn(0)
.1.3.6.1.4.1.19947.1.3.2.1.4.5 = BITS: 80 outputOn(0)
```

— get température globale

— nombre d'alims :

```
outputNumber OBJECT-TYPE
    SYNTAX  INTEGER (0..255)
    MAX-ACCESS  read-only
    STATUS  current
    DESCRIPTION
            "The number of output channels of the crate."

outputName OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (1..4))
    ACCESS  read-only
    STATUS  current
    DESCRIPTION
            "A textual string containing a short name of the
            output. If the crate is equipped with an alphanumeric
display, this string is shown to identify a output channel."

.1.3.6.1.4.1.19947.1.3.1.0 = INTEGER: 4
.1.3.6.1.4.1.19947.1.3.2.1.2.1 = STRING: +5V0
.1.3.6.1.4.1.19947.1.3.2.1.2.2 = STRING: +12V
.1.3.6.1.4.1.19947.1.3.2.1.2.4 = STRING: +3V3
.1.3.6.1.4.1.19947.1.3.2.1.2.5 = STRING: -5V2
```

— get température par alims : non sens, les capteurs ne sont pas répartis par alims.

```
outputMeasurementTemperature OBJECT-TYPE
    SYNTAX  INTEGER { OK (-128), FAILURE(127) }
    MAX-ACCESS  read-only
    STATUS  current
    DESCRIPTION
            "The measured temperature of the power module."

.1.3.6.1.4.1.19947.1.3.2.1.8.1 = INTEGER: OK(-128)
.1.3.6.1.4.1.19947.1.3.2.1.8.2 = INTEGER: OK(-128)
.1.3.6.1.4.1.19947.1.3.2.1.8.4 = INTEGER: OK(-128)
.1.3.6.1.4.1.19947.1.3.2.1.8.5 = INTEGER: OK(-128)

sensorNumber OBJECT-TYPE
    SYNTAX  INTEGER (0..8)
    ACCESS  read-only
    STATUS  current
    DESCRIPTION
            "The number of temperature sensors of the crate."

.1.3.6.1.4.1.19947.1.4.1.0 = INTEGER: 8

sensorTemperature OBJECT-TYPE
-- CHECK    SYNTAX  INTEGER { UNUSED(-128), (-127..127) }
    SYNTAX  INTEGER (-128..127)
    UNITS   "degres C"
    MAX-ACCESS  read-only
    STATUS  current
    DESCRIPTION
            "The measured temperature of the sensor.
```

```
 Unused temperature probes have the special value -128"

.1.3.6.1.4.1.19947.1.4.2.1.2.1 = INTEGER: 39 <B0>C
.1.3.6.1.4.1.19947.1.4.2.1.2.2 = INTEGER: 37 <B0>C
.1.3.6.1.4.1.19947.1.4.2.1.2.3 = INTEGER: 41 <B0>C
.1.3.6.1.4.1.19947.1.4.2.1.2.4 = INTEGER: 40 <B0>C
.1.3.6.1.4.1.19947.1.4.2.1.2.5 = INTEGER: 39 <B0>C
.1.3.6.1.4.1.19947.1.4.2.1.2.6 = INTEGER: 41 <B0>C
.1.3.6.1.4.1.19947.1.4.2.1.2.7 = INTEGER: -128 <B0>C
.1.3.6.1.4.1.19947.1.4.2.1.2.8 = INTEGER: -128 <B0>C
```

— get tension terminale par alims : non disponible

```
outputMeasurementSenseVoltage OBJECT-TYPE
    SYNTAX  Float
    UNITS   "V"
    MAX-ACCESS  read-only
    STATUS  current
    DESCRIPTION
            "The measured voltage at the sense input lines."

.1.3.6.1.4.1.19947.1.3.2.1.5.1 = Opaque: Float: 5.270000 V
.1.3.6.1.4.1.19947.1.3.2.1.5.2 = Opaque: Float: 12.169999 V
.1.3.6.1.4.1.19947.1.3.2.1.5.4 = Opaque: Float: 3.610000 V
.1.3.6.1.4.1.19947.1.3.2.1.5.5 = Opaque: Float: 5.670000 V
```

— tension constante max

```
outputSupervisionMaxSenseVoltage OBJECT-TYPE
    SYNTAX  Float
    UNITS   "V"
    MAX-ACCESS  read-write
    STATUS  current
    DESCRIPTION
            "If the measured sense voltage is above this value, the power supply
 performs the function defined by SupervisionAction."

.1.3.6.1.4.1.19947.1.3.2.1.17.1 = Opaque: Float: 5.450000
.1.3.6.1.4.1.19947.1.3.2.1.17.2 = Opaque: Float: 12.700000
.1.3.6.1.4.1.19947.1.3.2.1.17.4 = Opaque: Float: 3.650000
.1.3.6.1.4.1.19947.1.3.2.1.17.5 = Opaque: Float: 5.660000
```

— get ampérage par alims

```
outputMeasurementCurrent OBJECT-TYPE
    SYNTAX  Float
    UNITS   "A"
    MAX-ACCESS  read-only
    STATUS  current
    DESCRIPTION
            "The measured output current."

.1.3.6.1.4.1.19947.1.3.2.1.7.1 = Opaque: Float: 37.840000 A
.1.3.6.1.4.1.19947.1.3.2.1.7.2 = Opaque: Float: 3.090000 A
.1.3.6.1.4.1.19947.1.3.2.1.7.4 = Opaque: Float: 77.150002 A
.1.3.6.1.4.1.19947.1.3.2.1.7.5 = Opaque: Float: 83.769997 A
```

— put état (on/off)

```
sysMainSwitch OBJECT-TYPE
    SYNTAX  INTEGER { OFF (0), ON (1) }
    MAX-ACCESS  read-write
    STATUS  current
    DESCRIPTION
            "Read: An enumerated value which shows the current state of
 the crates main switch.
             Write: Try to switch the complete crate ON or OFF.
 Only the values ON or OFF are allowed."

.1.3.6.1.4.1.19947.1.1.1.0 = INTEGER: ON(1)
```

2.2 LibSnmp

2.2.1 Introduction

```
# yum install net-snmp-libs
# gcc -o go -lnetsnmp ?
# gcc -o go /usr/lib/libnetsnmp.so.15 ??
```

2.2.2 API SNMP

fichier *snmp.h* :

```
#ifndef __SNMP_H
#define __SNMP_H

#include <net-snmp/net-snmp-config.h>
#include <net-snmp/net-snmp-includes.h>

typedef struct snmp_session  Session;
typedef struct snmp_pdu      Pdu;
typedef struct variable_list List;       /* /usr/include/net-snmp/library/snmp_api.h:518 */

/* API */
Session* connectSnmp(char* ipAddress);      /* Warning: use 'admin' priviledges */
Session* disconnectSnmp(Session*);
Pdu*     getSnmp(Session*, char* requestedOid);
int      setSnmp(Session*, char* requestedOid, char types, char *values);

int      getASN_OCTET_STR(List*, char**);
int      getASN_INTEGER(List*);
float    getASN_OPAQUE_TAG2_FLOAT(List*);
/* int      getASN_BIT8(List*); */
/* int      getASN_BOOLEAN(List*); */

void     showAsnTypeForDebug(List* var);

#define boolError    2
#define intError    -32767
#define floatError  -1.
#define doubleError -1.

/* see exemple in section _utMAIN in snmp.c */

#endif /*  __SNMP_H */
```

2.2.3 API WIENER

fichier *wiener.h* :

```
#ifndef __WIENER_H
#define __WIENER_H

#include "global.h"
#include "snmp.h"

#define DEFAULT_IP_ALIM "192.168.1.86"
```

```
int setWienerOff(const char* ipAddress);
int setWienerOn(const char* ipAddress);

#endif  /*  __WIENER_H */
```

2.2.4 Fonctions WIENER

fichier *wiener.c* :

```
int setWienerOff(const char* ipAddress)
{
  Session* p_session = (Session*) 0;
  Pdu *response = (Pdu*)0;
  int rc = FALSE;

  if (ipAddress == (char*)0)
    ipAddress = DEFAULT_IP_ALIM;

  if ((p_session = connectSnmp(ipAddress))
      == (Session*)0)
    goto error;

  /* switch off */
  if (setSnmp(p_session, ".1.3.6.1.4.1.19947.1.1.1.0", 'i', "0\0")
      != TRUE)
    goto error;

  rc = TRUE;
 error:
  return rc;
}

int setWienerOn(const char* ipAddress)
{
  Session* p_session = (Session*) 0;
  Pdu *response = (Pdu*)0;

  /* outV, minV and MaxV */
  static char voltagesOids[][32] =
    {
      ".1.3.6.1.4.1.19947.1.3.2.1.10. ", /* outputVoltage */
      ".1.3.6.1.4.1.19947.1.3.2.1.16. ", /* minVoltage */
      ".1.3.6.1.4.1.19947.1.3.2.1.17. "  /* maxVoltage */
    };

  const int indexOfUnitsInVoltagesOids = 30;

  /* outV, minV and MaxV */
  float voltages[3];

  static char unit[]={'1', '2', '4', '5', '\0'};

  int try = 0;
  int nbConsecutiveValidatedTries = 0;
```

```
  int rcOnLoop = TRUE;
  int u, i;
  int rc = FALSE;

  if (ipAddress == (char*)0)
    ipAddress = DEFAULT_IP_ALIM;

  if ((p_session = connectSnmp(ipAddress))
      == (Session*)0)
    goto error;

  /* switch on */
  if (setSnmp(p_session, ".1.3.6.1.4.1.19947.1.1.1.0", 'i', "1\0")
      != TRUE)
    goto error;

  /* validate 3 time output voltage */
  do
    {
      /* loop on each unit */
      for (rcOnLoop = TRUE, u=0; rcOnLoop && unit[u] != '\0'; ++u)
      {
/* get outputVoltage, minVoltage and maxVoltage for unit u*/
for (i=0; rcOnLoop && i<3; ++i)
  {
    voltagesOids[i][indexOfUnitsInVoltagesOids] = unit[u];
    //printf("   %s\n", voltagesOids[i]);
    rcOnLoop &= ((response = getSnmp(p_session, voltagesOids[i])) != (Pdu*)0);
    rcOnLoop &= ((voltages[i] = getASN_OPAQUE_TAG2_FLOAT(response->variables)) != floatError);
    snmp_free_pdu(response);
  }

/* validate output voltage for this unit */
//printf("outputVoltage = %f < %f < %f\n", voltages[1], voltages[0], voltages[2]);
rcOnLoop &= (voltages[1] < voltages[0] && voltages[0] < voltages[2]);
      }

    /* validate output voltage once */
    if (rcOnLoop)
++nbConsecutiveValidatedTries;

    response = (Pdu *)0;
  }
  while(nbConsecutiveValidatedTries < NB_CONSECUTIVE_VALIDATED_TRIES_NEEDED
&& ++try < NB_MAX_TRY);

  rc = (nbConsecutiveValidatedTries == NB_CONSECUTIVE_VALIDATED_TRIES_NEEDED);
 error:
  return rc;
}
```

2.2.5 Code source

Je cherche à me déplacer dans les sous arbres, ce qui ne me semble pas trivial au regards des headers installés via le paquet **libsnmp-dev**.

A priori il n'y a rien de prévu à cet effet ? !

— fichier */etc/apt/sources.list* :

```
deb-src http://ftp.fr.debian.org/debian lenny main contrib non-free
```

— insatalation du code source de */usr/bin/snmpwalk*

```
# mkdir rangerLesSourcesIci && cd rangerLesSourcesIci
# apt-get source net-snmp
# less apps/snmpwalk.c
```

2.3 Analyseur grammatical

2.3.1 Introduction

2.3.2 Grammaire théorique

Il faut ajouter un nouveau shell : fichier */SBig/Server/Bigd/Big/POWERControl.c*. Sa grammaire sera du type :

```
order: POWER command1
     | POWER idAlim command2
     | POWER idSensor command3

idAalim: NE
       | NW
       | SE
       | SW

idSensor: S[1-6]

command1: ON
        | OFF
        | GET STATE
        | GET MAXTEMP

command2: GET SENSE VOLTAGE
        | GET NOMINAL VOLTAGE
        | GET CURRENT

command3: GET TEMP
```

2.3.3 Grammaire implémentée

```
order: POWER command1

command1: ON
        | OFF
```

2.4 Intégration

2.4.1 Introduction

Cette page décrits les modifications apportées au code source de BIG.

2.4.2 Bibliothèque SNMP

— répertoire */SBig/Server/include/net-snmp*
— fichier */SBig/misc/libnetsnmp.so* **note :** fichier binaire

2.4.3 Utilitaire WIENER

— fichier */SBig/Server/Bigd/include/Wiener.h*
— fichier */SBig/Server/Bigd/Util/Wiener.c*

2.4.4 Analyseur Grammatical

— fichier */SBig/Server/Bigd/Big/WIENERControl.c*
— fichier */SBig/Server/include/Control.h*

```
...
extern int  WIENERControl(char *);
...
```

— fichier */SBig/Server/Bigd/Big/Big.c*

```
...
if (!Look4Key(local_buffer,"WIENER",  6) && !(val = WIENERControl  (local_buffer+7))) return(NO
...
```

— fichier */SBig/Server/Bigd/Makefile*

```
...
LIBS   += -L../lib -lm -lpthread -lnetsnmp
...
$(PROGRAM):     $(wildcard Object/*.o)
                @echo -ne "Linking \t\033[1;31m$(PROGRAM)\033[0;39m ... \t "
                @[ -f ../lib/libnetsnmp.so ] || ln -s ../../misc/libnetsnmp.so ../lib/.
                ...
```

— fichier */SBig/Server/Client/WIENER_ON*

```
BEGIN
WIENER ON
ENDMSG
```

— fichier */SBig/Server/Client/WIENER_OFF*

```
BEGIN
WIENER OFF
ENDMSG
```

2.4.5 Test

— Compilation

```
$ cd ~/SBig/Server/Bigd
$ make
```

— Lancement du serveur

```
$ telnet 192.168.1.162
$ ~roche/SBig/Server/Bigd/big
```

— Allumer/éteindre l'alim 192.168.1.86

```
$ snmpwalk -v 2c -c public -On 192.168.1.86 .1.3.6.1.4.1.19947.1.1.1.0
.1.3.6.1.4.1.19947.1.1.1.0 = INTEGER: 0
```

```
$ telnet 192.168.1.162 1805
BEGIN POWER ON
ENDMSG

$ snmpwalk -v 2c -c public -On 192.168.1.86 .1.3.6.1.4.1.19947.1.1.1.0
.1.3.6.1.4.1.19947.1.1.1.0 = INTEGER: 1

$ cat WIENER_ON | ~/SBig/Server/Client/client 192.168.1.162 1805

$ snmpwalk -v 2c -c public -On 192.168.1.86 .1.3.6.1.4.1.19947.1.1.1.0
.1.3.6.1.4.1.19947.1.1.1.0 = INTEGER: 0
```

Chapitre 3

Châssis compact-PCI

Dans sa version finalisée, la caméra HESS2 sera équipée de 3 châssis PCI.

- Châssis **Acquistion**
  - segment **Slow Control** comprenant 1 cartes processeur (big) et 2 cartes **slc**.
  - segment **Fifo Data** comprenant 1 carte processeur (zora) et 4 cartes **fifo**.
- Châssis **Trigger**
  - 1 carte processeur (émilie)
  - 1 carte **trigger management**
  - 9 cartes **trigger** (1 pour chaque tranche de 11 secteur de la caméra)
- Châssis **Sécurité**
  - 1 carte processeur (bunny)
  - beaucoup de cartes en provenance du LAPP

Les carte processeur sont fournies par la société CES. Cette même société fournit le compilateur-croisé ELDK contenant aussi de la doc.

Voici l'énumération des cartes contrôleur en notre possession :

	Rev	module	rev	serial	host cpu	ver	speed	command	boot	desc
c10	5.4	4068 DE		425	PPC 7457	1.02	1000 Mhz	16/09/09	n1n9	spare
c11	5.4	4068 DE		426	PPC 7457	1.02	1000 Mhz	16/09/09	n1n9	spare
c12	5.4	4068 DE		423	PPC 7457	1.02	1000 Mhz		n1n41	zora
c13	5.4	4068 DE		424	PPC 7457	1.02	1000 Mhz		n1n41	émilie
c14	5.3	4065 AD	FE	2514	PPC 750	83.02	500 Mhz			
c15	4.9	4068 BE	EC	206	PPC 7455	3.04	800 Mhz		n1n9	locMod
c16	5.3	4068 DE	BA	270	PPC 7457	1.02	1000 Mhz		n1n41	big
c17	5.3	4068 DE	BA	271	PPC 7457	1.02	1000 Mhz		n1n41	bunny
c18	5.3	4068 DE	BA	273	PPC 7457	1.02	1000 Mhz		n1n9	testPm
c19	5.3	4068 DE	BA	272	PPC 7457	1.02	1000 Mhz		n1n9	camera

Sur ces 8 cartes, *c14* datant de HESS1 n'est plus utilisable et une *c15* s'est avérée incompatible sur un châssis partagé et aussi avec certains drivers.

3.1 Bus cPci

3.1.1 Introduction

Cette section décrit le fonctionnement des éléments à notre dispositions, qui s'articulent tous autour du bus COMPACT-PCI.

— Des châssis (plus communément appelés 'fond de panier' ou 'backplane') **compact pci** 21 slots. Les bus CPCI ne peuvent pas excéder 8 éléments. Aussi il s'agit de 3 châssis de 7 slots reliés par le biais de *bridges*. Ils sont de la marque *schroff*. Voici la documentation que j'ai trouvé.
— Des cartes contrôleur (documentation donnant accès à des outils de diagnostique et de configuration. Ce bios est accessible via un lien série.

Durant la phase de configuration du bus (p593) il y a 'adressages géographiques' (par slots) afin de déterminer des adresses qui seront ensuite utilisées vie 'adressage physique'. Cette phase d'initialisation consiste à lire et à compléter les premiers registres de chaque carte PCI que l'on nomme le BAR (p353) :

— Device ID et Vendeur ID (on utilise celui du Cern) Ils servent à connecter la carte via le driver générique *pci*.
— Subsystem ID et Vendeur ID Ils peuvent éventuellement venir compléter la recherche précédente.
— Base Address [0-5] (plages de mémoires. Par exemple sur les cartes FIFO il y a 2 plage : une de 128k et l'autre de 8k). Contient initialement le nombre de mémoire à allouer
Est écrasé avec l'adresse affectée.

En fait, le terme '*master*' désigne la carte qui est à l'initiative de l'échange. Le bus PCI ne peut avoir qu'un seul *arbitre* et il semble qu'il y ait une confusion des rôles qu'il va falloir éclaircir.

3.1.2 État des lieux (avant les patchs)

Par défaut, la carte contrôleur installée dans le premier slot du châssis (celui le plus à droite) est configurée comme *master*. Le deuxième contrôleur, installé dans n'importe quel autre slot sera alors définit en tant qu'*esclave*.

Procédure

Sur le contrôleur *master* la commande `diag cpci` nous permet de voir :

— l'adresse mémoire à laquelle les cartes esclaves pourront accéder à la mémoire du *master*.
— Le numéro des slots sur lesquels se trouvent les cartes esclaves.

Sur le contrôleur *slave* la commande `diag cpci` nous permet de voir :

— l'adresse mémoire à laquelle les autres cartes pourront accéder à la mémoire du contrôleur esclave.
— Le numéro du slots sur lesquels se trouve le contrôleur esclaves.

Remarque : le numéro localisant le contrôleur esclave retourné par les 2 contrôleurs ne correspond pas.

On peut tester l'accès mémoire de chacun des contrôleur. Lecture de la mémoire du *master* de le contrôleur *slave* :

```
master> pm.l 0
        ...
                                slave> dm.l <adresse master>
```

Lecture de la mémoire du *slave* de le contrôleur *master*.

```
                                slave> pm.l 0
                                ...
master> dm.l <adresse slave>
```

Résultats

La mémoire est indiqué en millions. Lorsqu'un chiffre figure (40, 42, 44...) cela signifie que la mémoire est indiqué explicitement et qu'elle est accessible par l'autre carte contrôleur conformément aux procédures décrites ci-dessus.

Numérotation des slots :

n. de droite à gauche	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
numérotation *master*								9	A	B	C	D	E	F		6	5	4	3	2	
numérotation *slave*	8	9	10	11	12	13	14	8	9	10	11	12	13	14	1	2	3	4	5	6	

Adresses mémoire affectées avec 2 cartes contrôleur en fonction de l'emplacement choisi pour la carte *slave* :

slot *slave*	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	aucun
adresse *master*	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	42	42	42	42	42	40
adresse *slave*								42	42	42	42	42	42	42		40	40	40	40	40	

Adresses mémoire affectées avec 2 cartes contrôleur et jusqu'à 2 cartes fifo :

21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	aucun
																			F	40
																		40	F	42
																	F	40	F	42
													F				40			42
																40	F		F	42
													F			40			F	42
												F	F			40				42
											40	F	F							44
											40		F							44
										42									F	40

FIGURE 3.1 – représentation du châssis 21 slots

Conclusion

— On peut penser que les numéro de slot *master* et *slave* discordent mais n'empêchent pas les cartes de communiquer.
— Les cartes contrôleur sont interchangeables sans que les adresse *master* et *slave* changent.
— Les contrôleurs *slave* sur les slots 14 à 21 peuvent accéder à la mémoire *master* mais pas l'inverse.
— Les adresses mémoires affectées change en fonction de la disposition des cartes sur le châssis.

S'agit-il de bridge transparents ou non ?

3.1.3 Mapping depuis le second slot (patch 1)

Nous souhaitons que la carte contrôleur qui n'est pas dans le premier slot puisse récupérer les ressources de configuration CPCI.

— Patcher le noyau. Étrangement, il n'y a pas de soucis avec le problèmes annoncé. Cf le mail de **Rupert** :

```
Normally the CES-provided kernel will only enumerate if it detects itself
as system slot. However this can be changed easily in the file
arch/powerpc/platforms/cesrio/pci.c, by writing a 1 into
cpci_cfg->sys_slot in the function rio_find_bridges, before the loop where
rio_init_pci_bus() is called. We may plan to make this configurable in the
next release.

Because normal linux PCI device drivers occupy all devices they find,
it would lead to problems if the same device is found by two rios. This
is the reason why only the master slot enumerates.

To work around that problem, newer SugarHat releases (I think since 3.2.0 or
3.2.1) support filtering PCI slots, in a way that they are invisible to the
operating system. We have implemented this for other boards
with two processors, but is should be also usable in your case.
```

— Compiler puis booter le noyau avec l'extension *-shl-3.2.1* car elle est vérifiée lors du chargement des drivers *papat26**.
— Compiler puis charger le module *xpc* et *xpc_dma*

```
$ telnet 192.168.1.158
# cd /usr/src/xpc/driver
```

```
# make
# make install
# depmod -r
# modprobe -l
# modprobe xpc
# ls /proc/bus/xpc/
cpci  lpci  xpci
# modprobe xpc_dma
```

— Charger les modules

```
# insmod /home/camera/modules/papat26_dat.ko
# insmod /home/camera/modules/papat26_slc.ko
```

Attention : les fond de tiroirs PCI peuvent recevoir des cartes des 2 côtés. On peut endommager les cartes en les branchant dans un slot libre faisant face à un slot occupé.

3.1.4 Mapping derrière le second bridge (patch 2)

Les derniers slots ne sont pas mappés. Cela ce voit par l'absence des allocation mémoires énumérées par la commande `lspci -vv`. Ici, la 3ème carte n'es pas mappée :

```
# lspci -vv
0001:80:0b.0 Class 1180: 10dc:2ccc
        Control: I/O- Mem+ BusMaster- SpecCycle- MemWINV- VGASnoop- ParErr- Stepping+ SERR- FastB2B-
        Status: Cap- 66Mhz- UDF- FastB2B+ ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR-
        Interrupt: pin A routed to IRQ 32
        Region 0: Memory at 0000000042000000 (32-bit, non-prefetchable) [size=128K]
        Region 1: Memory at 0000000042100000 (32-bit, non-prefetchable) [size=8K]

0001:80:0e.0 Class 1180: 10dc:2ddd
        Control: I/O- Mem+ BusMaster- SpecCycle- MemWINV- VGASnoop- ParErr- Stepping+ SERR- FastB2B-
        Status: Cap- 66Mhz- UDF- FastB2B+ ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR-
        Interrupt: pin A routed to IRQ 29
        Region 0: Memory at 0000000042200000 (32-bit, non-prefetchable) [size=128K]
        Region 1: Memory at 0000000042300000 (32-bit, non-prefetchable) [size=8K]

0001:88:0c.0 Class 1180: 10dc:2ccc
        Control: I/O- Mem- BusMaster- SpecCycle- MemWINV- VGASnoop- ParErr- Stepping+ SERR- FastB2B-
        Status: Cap- 66Mhz- UDF- FastB2B+ ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR-
        Interrupt: pin A routed to IRQ 31
```

Voici le patch de rupert. Il faut extraire l'archive directement dans les sources du noyau.

3.1.5 État des lieux (après les patchs)

Détails de la commande `lspci` :

```
# lspci
00:01.0 Class 0200: 1022:2000 (rev 44)
00:02.0 Class 0200: 8086:1008 (rev 02)
0001:20:04.0 Class 1180: 10dc:2bbb
0001:20:08.0 Class 0604: 3388:0026 (rev 04)
0001:80:08.0 Class 0604: 3388:0026 (rev 04)
0001:80:0e.0 Class 1180: 10dc:2aaa
```

La commande `lspci` fournit les détails suivants :

— bus et slot : [0001] :bb :ss.O
— class : xxxx
— id : xxxx :xxxx
— identifiant : `xxxx:xxxx`

id	carte
3388 :0026	bridge
1022 :2000	ctrl slot1
8086 :1008	ctrl slot2 (vu depuis ctrl1)
10d9 :4065	ctrl slot2 (vu depuis ctrl2)
10dc :2aaa	slow controle
10dc :2bbb	fifo data
10dc :2ccc	triger
10dc :2ccc	triger management

— domaine et bus `[dddd]:bb:xx.O`
domaine : rien=LPCI, 0001=CPCI
bus :

00	domaine LPCI (premier segment)
20	1er segment
80	2ème segment (derrière le 1er bridge)
88	3ème segment (derrière le 2ème bridge)

— slot `[0001]:xx:ss.O`
Le châssis est composé de 3 segments de 7 slots numérotés différemment :

de 0x01 à 0x07	1er segment
de 0x0f à 0x09	2ème segment (derrière le 1er bridge)
de 0x0f à 0x09	3ème segment (derrière le 2ème bridge)

remarque : le bridge est affecte au slot 8 que ce soit pour le premier ou le deuxième segment.
remarque : on obtient le même chiffrage a partir du premier slot que ce soit sur un châssis de 14 ou de 21 slots.

```
PCI-TREE
├─LPCI Bridge BUS 0x00
│ ├─0000:00:01.0
│ └─0000:00:02.0
└─CPCI Bridge BUS 0x20
  ├─0001:20:01.0
  ├─...
  └─0001:20:07.0
    └─0001:20:08.0 BUS 0x80
      ├─0001:80:08.0 BUS 0x88
      │ ├─0001:88:09.0
      │ ├─...
      │ └─0001:88:0f.0
      ├─0001:80:09.0
      ├─...
      └─0001:80:0f.0
```

FIGURE 3.2 – *liste des devices par Linux*

3.1.6 Nouveau code source shl-3.2.3

— don't boot

```
Please try to append "cpci_full_enum" to the boot_line. I had to
disable this feature by default because it is not useful to all
customers.

> By using the new kernel source, I cannot boot on the first slot, with or
> without a second controller card in the second slot. However there is no
> problem booting on the second slot.
```

— mapping pci
```
Please try the file in the attachment. It replaces arch/powerpc/kernel/pci_32.c.

> 'lspci -vv' shows :
> Region 0: Memory at <ignored> (32-bit, non-prefetchable)
```
— 2 Processeurs sur les slots 1 et 2 :
Lorsqu'il y a des cartes au delà du premier segment, il faut ajouter `cpci_full_enum` aux paramètres du noyau du processeur en slot 1.
— 2 Processeurs sur les slots 1 et 2 avec un châssis 14 slots :
Les cartes ne charge pas leur firmware et clignotent tous les deux en bleu.
— 2 Processeurs sur les slots 1 et 8 :
Il faut recompiler le noyau en commentant le slot 8 dans le fichier *arch/powerpc/platforms/cesrio/rio3Resources.cesRT*.
Le résultat est que le processeur en slot 1 ne peut plus franchir le premier bridge.

3.1.7 Carte rio sur le second segment (patch petit nicolas)

Code ajouté au fichier *arch/powerpc/platforms/cesrio/pci.c* :

```
/***** Add for Hess2 *****/

static int slaveInsertedOnSecondSegment = 0;
static int __init parse_slaveInsertedOnSecondSegment(char *arg)
{
        slaveInsertedOnSecondSegment = 1;
        return 0;
}
early_param("seg_2", parse_slaveInsertedOnSecondSegment);

static void fixup_hess2_enable2ndBridge(struct pci_dev *dev) {
  if (slaveInsertedOnSecondSegment){
    printk("RIO_PCI: %s call by slave rio on 21 slot crate\n", __func__);

    pci_slot_info[2][1].visible=1;      // enable device
  }
}
DECLARE_PCI_FIXUP_EARLY(0x8086, 0x1008, fixup_hess2_enable2ndBridge);

static void fixup_hess2_dontMapSlave(struct pci_dev *dev) {
  printk("RIO_PCI: %s call by master rio on 21 slot crate\n", __func__);

  dev->class = PCI_CLASS_NOT_DEFINED;  // don't map device
  dev->hdr_type = 3;                   // free device
}
DECLARE_PCI_FIXUP_EARLY(0x10d9, 0x4065, fixup_hess2_dontMapSlave);

/* end Add for Hess2 *****/
```

- Modifications du comportement du noyau :
  - L'esclave inséré sur le segment 2 est capable de voir le second bus (utile seulement si des cartes sont présentes sur le 3ème segment).
  - Le maître ne mappe plus la carte esclave afin de voir les même carte que l'esclave a partir du deuxième segment.
- Résultats :
  - non régression : le châssis fonctionne (comme avant) avec les 2 rios dans les premiers slots.
  - Les 2 premiers segments du châssis fonctionnent (drivers, big et zora) bien avec l'esclave sur le second segment.
  - Il semble que le 3ème segment soit correctement mappe lorsque l'esclave boot avec au moins une seconde de retard sur le second segment. Cependant, cela frise parfois complètement le maître lors du montage NFS ou du login.
  - rq : les interruption ne sont pas convenablement reparties.
  - rq : les cartes du 2ème segment sont numérotées dans le sens oppose par les 2 rios : il ne vaut mieux pas y melanger les types de cartes "a" et "b".
- Tests :
  - l'esclave "B" boot une seconde apres le maitre "A".
  - le maitre "A" charge le driver des cartes "a".
  - l'esclave "B" charge le driver des cartes "b".
  - Tests de non regression :
    - options du noyau de la rio maitre :
      `boot_line : cpci_full_enum`
    - options du noyau de la rio esclave :
      `boot_line : ip=:::::eth1 cpci_full_enum`

21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01
									b	b	b	b					a	a	B	A
			a	b						b	b	b					a		B	A
			a	b						b	b	b					a	B		A
			a	b						b	b	b				B	a			A
				b								b	a	b	B	b	a			A

  - — Tests pour la camera 20 tiroirs :
    - options du noyau de la rio maitre :
      `boot_line : cpci_full_enum`
    - options du noyau de la rio esclave :
      `boot_line : ip=:::::eth1 cpci_full_enum seg_2`

21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01
												b	B						a	A
											b		B						a	A
										b			B						a	A
									b				B						a	A
								b					B						a	A
							b						B						a	A
							b					B							a	A

— — Tests pour la grande camera :
  — options du noyau de la rio maitre :
    `boot_line : cpci_full_enum`
  — options du noyau de la rio esclave :
    `boot_line : ip=:::::eth1 cpci_full_enum seg_2`

21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01
								b	b	b			B					a	a	A
								b	b	b	b		B				a	a		A
							b	b	b	b			B				a	a		A
							b	b	b	b			B			a	a			A
							b	b	b	b			B			a	a			A
							b	b	b	b		B					a	a		A
							b	b	b			B	b				a	a		A
							b	b			B	b	b			a	a			A
							b	b		B		b	b			a	a			A
							b		B		b	b	b			a	a			A
								B		b	b	b	b			a	a			A
							B			b	b	b	b			a	a			A
	a						B			b	b	b	b	a						A
	a						b	b	b	b		B						a		A
	a							b	b	b	b		B					a		A
						b	b				b	b	B				a	a		A
						a	b			b	b	b	B					a		A

— — Cas douteux :
  — options du noyau de la rio maitre :
    `boot_line : cpci_full_enum`
  — options du noyau de la rio esclave :
    `boot_line : ip=:::::eth1 cpci_full_enum seg_2`

21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01
							a	b	b	b	b		B					a		A

3.2 Compilation croisée

3.2.1 Introduction

To install SHL 3.2.0 on your host machine, you first have to insert the Sugar Hat Linux 3.2.0 CD into your host machine's CD-ROM drive. The host machine is typically a PC running a recent version of Linux. The full installation requires approximately 500 MiB of disk space.

3.2.2 Test des outils CES sous DEBIAN

Bien que le processus d'installation soit basé sur le principe des paquets RPM, ils sont indépendants de ceux éventuellement disponibles sur la distribution utilisée.

Installation des outils et d'ELDK

```
# apt-get install dialog
# mount /cdrom
# cp -fr /cdrom/* eldk

# cd eldk
# cp scripts/defaultconfig  .defaultconfig
# cp scripts/txtconfig .txtconfig
# cp scripts/menuconfig .menuconfig
# ./shl-install
```

If you installed in a location other than the default, replace /opt/CES/shl-3.2.0 in the path below with the directory that you used.

- *—/opt/CES/shl-3.2.1/eldk-4.0/usr* Contains the cross development toolchain
- *—/opt/CES/shl-3.2.1/rootfs-eldk-ppc_6xx* Contains the NFS-mountable root file system for 6xx and 7xx processors.
- *—/opt/CES/shl-3.2.1/rootfs-eldk-ppc_74xx* Contains the NFS-mountable root file system for 74xx processors.

Cross-compilation du noyau LINUX

La partition racine contient le noyau *arch/ppc/boot/images/zImage.rio*.

```
# . /opt/CES/shl-3.2.1/sugarhatlinux-3.2.1-env-eldk-ppc_74xx.sh
# cd /opt/CES/shl-3.2.1/eldk-4.0/ppc_74xx/usr/src/linux-2.6.24.7-rt13-shl-3.2.1
# make clean
# make zImage
# make modules
# make modules_install
```

Attention, `make mrproper` **efface la pré-configuration du noyau**

La partition racine */opt/CES/shl-3.2.1/eldk-4.0/ppc_74xx*
contient le noyau *arch/ppc/boot/images/zImage.rio /tftpboot/zImage*.

Exportation des fichiers sur N1N9 :

```
# cd /opt/CES/shl-3.2.1/eldk-4.0/ppc_74xx
# tar -zcf /tmp/racineNroche.tgz .
# scp /tmp/racineNroche.tgz root@192.168.1.9:/opt/sugarhat/devkit/targetNroche/racine.tgz
```

3.2.3 Cross-compilation sur N1N9

test de Cross-compilation du noyau LINUX

```
# cd /opt/sugarhat/devkit/
```

```
# export PPCDIR=ppc_74xx_nroche
# . ./sugarhat-env-current.sh

# cd targetNroche/usr/src/linux
# make zImage
```

Installation d'ELDK

```
$ ssh root@n1n9
# cd '/data/CES/CDX 69840L 3.2.1'
# ./shl-install

INSTDIR        [/tmp/ces.090528]
INST_KERN      [y]
INST_ELDK_74xx [y]
INST_MFCC      [y]
INST_XPC       [y]
INST_BPNET     [y]

# mv /tmp/ces.090528/eldk-4.0/ppc_74xx /opt/sugarhat/devkit/eldk/ppc_74xx_20090528
# rm -fr /tmp/ces.090528

# cd /opt/sugarhat/devkit/eldk/
cat >> LISMOI.TXT
ppc_74xx_20090528 : version originale extraite via /data/CES/CDX 69840L 3.2.1/shl-install le 28/05/2
^D

# cp -Rp ppc_74xx_20090528 ppc_74xx_20090528_c12
```

Configuration du système d'ELDK

Attention : ici nous faisons abstraction des liens symboliques :
— Pensez à utiliser `# tar` ou `cp -Rp` pour duppliquer les arborescences.
— Pensez à mettre à jour l'export des partitions NFS.

```
# . sugarhatlinux-3.2.3-env-eldk-ppc_74xx.sh
# cd eldk
# cp ppc_74xx ppc_74xx_fromScratch
# cd ppc_74xx/usr/src/linux
# make
# make modules
# make modules_install
```

Remarque : pensez à ajouter l'export NFS des */home* via le fichier */etc/fstab.local* comme décrits dans la rubrique 'outils'.

Conformément au fichier */etc/rc.sysinit* :

```
293:for file in /etc/sysconfig/modules/*.modules ; do
```

créer sur les partition racines des caméras adéquates le fichier */etc/sysconfig/modules/rpc.modules* :

```
modprobe xpc
```

Note : Il faut recompiler le noyau sur la cible afin de pouvoir compiler les drivers (sur cible).

cross-compilation du noyau LINUX

```
# ssh root@n1n9
# cd /opt/CES/shl-3.2.3

# export SHL_ROOTDIR="/opt/CES/shl-3.2.3/eldk-4.0/ppc_74xx_c14"
# export SHL_CROSSCOMPILER="/opt/CES/shl-3.2.3/eldk-4.0/usr/bin/ppc_74xx-"
# export SHL_INCDIR="/opt/CES/shl-3.2.3/eldk-4.0/ppc_74xx_c14/usr/include/ces"
# export ARCH="powerpc"
# export CROSS_COMPILE="/opt/CES/shl-3.2.3/eldk-4.0/usr/bin/powerpc-linux-"
# export INSTALL_MOD_PATH="/opt/CES/shl-3.2.3/eldk-4.0/ppc_74xx_c14"
# export TOPDIR="/opt/CES/shl-3.2.3/eldk-4.0/ppc_74xx_c14/usr/src/linux-2.6.24.7-rt21-shl-3.2.3"
# export SHL_KERNELSRC="/opt/CES/shl-3.2.3/eldk-4.0/ppc_74xx_c14/usr/src/linux-2.6.24.7-rt21-shl-3.2
# export KERNEL_SRC="/opt/CES/shl-3.2.3/eldk-4.0/ppc_74xx_c14/usr/src/linux-2.6.24.7-rt21-shl-3.2.3"

# cd target14/usr/src/linux
# make menuconfig
```

3.2.4 Cross-compilation sur N1N40

Sur les machines 64 bits, il faut préalablement installer la glibc 32 bits :

```
# yum install glibc.i686
```

Puis réinstaller le cross compilateur.

3.2.5 Cross-compilation sur UBUNTU

Installation d'ELDK

```
# cd /media/sda2/hess/isoCES
# mount CDX69840L_3.2.3.iso -o loop /cdrom
# cp -fr /cdrom/* .
# umount /cdrom

# chmod +x scripts/txtconfig
# ./shl-install

# cd /opt
# ln -s CES sugarhat
# cd sugarhat
# ln -s shl-3.2.3 devkit
# ls /opt/sugarhat/devkit/eldk-4.0/usr/bin/ppc_74xx-*
```

Cross compilation de BIG

```
# adduser roche
# cd /home
# rm -fr roche
# ln -s /opt/CES/home/roche roche
# chown -R roche.roche roche/*
# su roche

$ cd ~/SBig/Server
$ make clean
$ make
```

3.2.6 Compilation sur cible (carte contrôleur)

Ajout de la bibliothèque NET-SNMP

Il s'agit d'installation sur une distribution LINUX FROM SCRATCH et par conséquent, bien que l'outils RPM soit disponible, aucun paquet n'est installé. Par ailleurs l'installation d'un meta-gestionnaire de paquet (APT) necéssite préalablement l'installation de WGET, elle même assez compliquée. Bien que l'on pourrait copier les fichiers depuis une autre machine POWER-PC, nous n'en avons pas à dipsosition. Par conséquent, il semble que la seule alternative viable soit la compilation. L'utilisation du cross-compilateur n'étant pas nécessaire, autant se simplifier la tâche en compilant directement sur une carte controleur.

— *net-snmp-5.4.2.1*
Le fichier *FAQ* est intéressant. Version light :

```
$ ./configure --disable-agent --disable-applications --disable-manuals --disable-scripts \
  --disable-mibs --disable-mib-loading | tee config.log2

Default version of SNMP to use : 3
System Contact Information : nicolas.roche@lpnhe.in2p3.fr
System Location : rioc4068
Location to write logfile (/var/log/snmpd.log):
Location to write persistent information (/var/net-snmp):

$ make
# make install

$ ls /usr/local/lib/
libnetsnmp.a   libnetsnmp.so     libnetsnmp.so.15.1.2
libnetsnmp.la  libnetsnmp.so.15

$ ls /usr/local/include/net-snmp/
agent          machine              output_api.h   system         version.h
config_api.h   mib_api.h            pdu_api.h      types.h
definitions.h  net-snmp-config.h    session_api.h  utilities.h
library        net-snmp-includes.h  snmpv3_api.h   varbind_api.h
```

— fichier */etc/ld.so.conf*

```
/usr/local/lib
```

— configure dynamic linker run-time bindings

```
# ldconfig
```

— **remarque** : pensez a copier la librairie dans l'environnement de cross-compilation.

```
$ cp /usr/local/lib/libnetsnmp.so.20.0.0 /home/roche/SBig/misc/libnetsnmp.so.20
$ vi /home/roche/SBig/Server/Bigd/Makefile
@[ -f ../lib/libnetsnmp.so ] || ln -s ../../misc/libnetsnmp.so.20 ../lib/libnetsnmp.so
```

compilation du noyau LINUX

```
# cd /usr/src/linux
# make menuconfig
# make && make modules_install

# cd /usr/src/xpc/drivers
# make
# make install
```

3.2.7 Debuggage

Sisi c'est possible.

Console

```
                                        $ telnet camera16
                                        camera16$ cd /home/roche/SBig/Server/Bigd/big
                                        camera16$ gdbserver host:1234 big

$ cd /home/roche/SBig/Server/Bigd (important)
$ /opt/sugarhat/devkit/eldk-4.0/usr/bin/ppc_74xx-gdb big
(gdb) target remote camera16:1234
Remote debugging using camera16:1234
0x30010604 in ?? ()   # objdump -h show that 0x30010604 is outside the program
(gdb) b main
Breakpoint 1 at 0x1001446c: file Server.c, line 494.
(gdb) c               # At this point the program counters is *inside* (do not use r!)
(gdb) info threads    # énumérer les threads
...
(gdb) thread 2        # changer de thread
...
(gdb) quit
```

Remarque : celà marche aussi avec EMACS :

Emacs

```
                                        $ telnet camera16
                                        camera16$ cd /home/roche/SBig/Server/Bigd/big
                                        camera16$ gdbserver host:1234 big
C-f /home/roche/SBig/Server/Bigd (important)
M-x gdb
Run gdb (like this): /opt/sugarhat/devkit/eldk-4.0/usr/bin/ppc_74xx-gdb --annotate=3 big
(gdb) target remote camera16:1234
Remote debugging using camera16:1234
0x30010604 in ?? ()
(gdb) c
```

FAQ

— Parfois on perd le code source (à priori) à cause de la compilation récursive. Pour récupérer les fichier source, il faut ajouter les répertoires concernés au pseudo `PATH` de GDB.

```
(gdb) directory Server
Source directories searched: /mnt/homes/lpnp90/roche/cvs/SBig/Server/Bigd/Server:$cdir:$cwd
```

— Si le curseur saute d'une ligne à l'autre en revenant sur ces pas, alors il faut minimiser l'optimisation du compilateur dans le fichier *Make.rules* :

```
#CFLAGS         += -O6
CFLAGS          += -g -O1
```

3.3 Boot des cartes contrôleur

3.3.1 Introduction

Voici la documentation.

3.3.2 Booter depuis le firmware PPCMon

Valeurs d'origines :

inet_host	IP number of this target	192.168.1.168
inet_bplane	IP on BpNet (optional)	0.0.0.0
inet_bootserver	IP number of TFTP and NFS server	192.168.1.9
inet_gateway	Default gateway (optional)	192.168.1.3
inet_nameserver	Nameserver : DNS (optional)	192.168.1.3
inet_protocol	do we use the BOOTP protocol	arpa
inet_mount	NFS mount point on server machine	192.168.1.9 :/opt/CES/devkit/target8
inet_ethermode		auto

boot_flags	Not used to by Linux	a
boot_device	Boot from local ethernet	le
boot_filename	Filename in tftp for the kernel image	/tftpboot/zImage8
boot_rootfs	Mount NFS root filesystem	nfs
boot_delay	Should be > 0	3 sec
boot_address	Load address should be a multiple of 16kB	1000000
boot_size	Memory used by Linux (0=all)	0
boot_fast	Must be disabled	n
boot_filetype	Must be 'binary' for zImage	binary
boot_cpunr	Required on VME : Node id for bpNet	0
boot_mode	ppcmon VS auto	auto
boot_line	Linux kernel parameters	ex : ip= : : : : :eth1

3.3.3 Configurations particulières

C18 : banc de test Corona

inet_bootserver	192.168.1.9
inet_mount	192.168.1.9 :/opt/sugarhat/devkit/target18

boot_filename	/tftpboot/zImage-RIOC4068
boot_delay	3 sec
boot_line	

C19 : module 20 tirroirs

inet_bootserver	192.168.1.9
inet_mount	192.168.1.9 :/opt/sugarhat/devkit/target19

boot_filename	/tftpboot/zImage19cameraTest
boot_line	

3.3.4 Utilisation des drivers trigers

Lors du chargement des drivers des cartes triggers, on obtient un message d'erreur relatif aux paramètres. Si l'on creuse avec 'dmesg' précise qu'une erreur de collision empêche le driver 'trigger' de se charger. Pour y remédier, il faut positionner le paramètre `diag_cpci` à `y`.

```
ppcmon> set diag_cpci
--> y
```

3.3.5 Utilisation de cartes à partir du 2ème segment

Pour booter avec des cartes au delà du premier segment :

```
ppcmon> set diag_cpci
--> y
ppcmon> set diag_xpci
--> y
```

3.3.6 Plusieurs cartes processeurs

Pour booter avec 2 cartes controleur, si des cartes étant présentes au delà du premier segment alors il faut ajouter l'option du noyau `cpci_full_enum` sinon la deuxième carte refuse de booter. Par ailleurs cette option est aussi requise pour l'unique processeur du châssis trigger.

```
ppcmon> set boot_line
--> cpci_full_enum
```

Remarque : On ne sait pas encore bouter avec des cartes controleur au delà du premier segment du châssis 21 slot.

3.3.7 Charger les drivers au démarage

Toutes les opération décrites ci-dessous se passent sur cible.

— compiler les drivers

```
$ telnet camera16
# su roche
$ cd ~/SBig/Driver/TmgTs/
$ make
tmgts.h:20:26: error: linux/config.h: No such file or directory
tmgts.h:25:27: error: linux/wrapper.h: No such file or directory
```

— supprimer les inclusion des fichier *linux/config.h* et *linux/wrapper.h*

```
$ make
```

— ranger les drivers dans */lib/modules/2.6.24.7-rt21-shl-3.2.3*

```
# mkdir /lib/modules/2.6.24.7-rt21-shl-3.2.3/papat2.6
# cp ~roche/SBig/Driver/TmgTs/tmgts.ko /lib/modules/2.6.24.7-rt21-shl-3.2.3/papat2.6/
```

— mettre à jour la liste de dépendance des modules */lib/modules/.../modules.dep*

```
# depmod -a
# modprobe tmgts
# modprobe -r tmgts
```

— créer le fichier */etc/sysconfig/module/xxx.modules* avec les droits d'execution.

```
# grep -n 'sysconfig/modules' -B1 -A2 /etc/rc.sysinit
151-# Load other user-defined modules
152:for file in /etc/sysconfig/modules/*.modules ; do
153-  [ -x $file ] && $file
154-done

# cat > /etc/sysconfig/modules/emilie.modules << FIN
> modprobe trig_drv26
> modprobe tmgts
> FIN
```

```
# chmod +x /etc/sysconfig/modules/emilie.modules
# /etc/sysconfig/modules/emilie.modules
# lsmod
# reboot
# lsmod
```

En fait, mieux vaut utiliser le fichier */etc/rc.local* pour les drivers développés car il permet de charger des drivers depuis les */home* montés via NFS.

3.4 Carte ethernet gigabit

3.4.1 Introduction

Il s'agit d'utiliser un des deux slot PCI présent sur les cartes compact-PCI.
— 192.168.8.1 Carte controleur
— 192.168.8.2 Pizza Box
— 192.168.8.3 Pc Nec

3.4.2 Modules

Carte controleur

La carte mise à disposition est une carte Intel(R) PRO/1000 Gigabit et son driver est le module *e1000*.
— Carte muette.
Bien que le module **e1000** se charge et que la carte se configure normalement, la carte n'émet aucune requête ARP et semble ne pas recevoir de trames non plus. Le symptôme est qu'il manque le flag `RUNNING` dans le compte rendu de **ifconfig**. En ajoutant la définition de la variable DBG (`#define DBG`) au début du fichier */usr/src/linux/drivers/net/e1000/e1000_osdep.h* on constate que le lien ne peut être établi (`Unable to establish link!!!`). Pour contourner cette erreur, nous avons flashé l'EEPROM de la carte GIGABIT :

```
# modprobe e1000
# ~/ethtool-6/ethtool -E eth1 magic 0x10088086 offset 31 value 0xf2
```

L'outils ETHOOL utilise en fait le driver de la carte éthernet. Le module utilise le numéro `magic` pour vérifier l'identité de la carte à modifier (cf *e1000/e1000_ethtool.c*). Ce numéro correspond aux numéros identifiants du vendeur et de la carte.

```
# lspci -n
00:02.0 Class 0200: 8086:1008 (rev 02)
```

— Boot.
Si l'on compile le driver en dur dans le noyau, alors au boot le noyau n'arrive pas à monter la partition racine via NFS. On dirait que le noyau inverse les interfaces **eth0** et **eth1** au cours du processus de bout. Etrangement, que l'on boot sur l'une interface ou l'autre, le boot s'arrête toujours au montage NFS.
Si l'on compile le driver en tant que module, alors le boot se passe normalement. cf */etc/rc.d/rc.sysinit :152* :

```
echo "modprobe e1000" > /etc/sysconfig/modules/e1000.modules
chmod +x /etc/sysconfig/modules/e1000.modules
```

Avec le noyau monolitique on s'en sort en modifiant les paramètres de boot depuis PPCMON.

```
PPCMon> set boot_line
 -> ip=:::::eth1
```

Cf le mail de **Rupert** :

```
If you have a PMC ethernet controller plugged, the on-board ethernet will
be enumerated later because its driver is loaded later.
To boot from on-board ethernet we use the boot_line argument "ip=::::eth1" to
use the other device for boot-time IP configuration.
```

Pizza Box

La carte utilisée n'est pas référencée sur le site d'Intel.

```
[root@lpnhess211 sysconfig]# lspci | grep Ethernet
1e:00.0 Ethernet controller: Intel Corporation 80003ES2LAN Gigabit Ethernet Controller (Copper) (rev
1e:00.1 Ethernet controller: Intel Corporation 80003ES2LAN Gigabit Ethernet Controller (Copper) (rev

# lspci -n | grep 1e:00.
1e:00.0 0200: 8086:1096 (rev 01)
1e:00.1 0200: 8086:1096 (rev 01)
```

Elle utilise le module **e1000e**.

```
# ethtool -i eth1
driver: e1000e
version: 0.3.3.3-k6
firmware-version: 1.0-11
bus-info: 0000:1e:00.1
```

PC Nec

Realtek RTL8168 Gigabit Ethernet controller

```
# lspci | grep Gigabit
03:00.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL8111/8168B PCI Express Gigabit Ether

# lspci -n | grep 03:00.0
03:00.0 0200: 10ec:8168 (rev 01)
```

Elle utilise le module **r8169**.

```
# ethtool -i eth0
driver: r8169
version: 2.2LK-NAPI
firmware-version:
```

3.4.3 Configuration

Cf le mail de **Rupert** :

```
We have achieved around 600-700 MBit with RIO3 hardware (Which should
be similar to your RIOC), but it requires some fine-tuning to achieve
such performance. I will try to find the old docs, but from my memory,
you can try the following:
- Set prefetch size to maximum
  You can set it in PPC_Mon with
  set bridge_lpci_prefetch 32
  (replace lpci with cpci if you have the card on cpci)
  you can verify the correct setting in linux when the xpc driver
  is loaded with
  cat /proc/bus/xpc/lpci/info
- use jumbo frames
  you can alter the mtu in linux with
  ifconfig eth0 mtu 9000
  note that you need to set the same mtu on all network interfaces on that
  network and you switches need to support it (they usually do).
- cache flushing
  This is already part of the standard CES kernel, and you don't need to
  activate it.
  We do an explicit cache flush before data written by the CPU is read by a
  PCI device. This avoids retries on the XPC bus which are very costly on the
  RIO hardware.
```

Note : Afin de ne pas dégrader les performances, ne pas placer la passerelle par défault sur l'interface gigabit.

Carte controleur

Via le shell PPCMON :

```
> set bridge_lpci_prefetch 32
> boot
# modprobe xpc
# cat /proc/bus/xpc/lpci/info
```

Fichier */etc/sysconfig/network-scripts/ifcfg-eth0* sur la carte controleur.

```
DEVICE=eth0
IPADDR=192.168.8.1
NETMASK=255.255.255.0
NETWORK=192.168.8.0
BROADCAST=192.168.8.255
ONBOOT=yes
NAME=gigabit
MTU=9000
```

Nous avons essayé d'utiliser la même interface sur la pizza box et donc une seconde IP sur le même réseau depuis zora :

Fichier */etc/sysconfig/network-scripts/bonnet-bleu* sur la carte controleur.

```
if [ $1 == "down" ]
then

  route del -host 192.168.1.40 dev eth0
  route del -host 192.168.1.41 dev eth0

else

  route del -net 192.168.1.0 netmask 255.255.255.0 dev eth0
  route del -net 169.254.0.0 netmask 255.255.0.0 dev eth0
  route add -host 192.168.1.40 dev eth0
  route add -host 192.168.1.41 dev eth0

fi
```

En fait les route ne sont d'aucune utilité car ici nous n'avons pas configuré le noyau pour agir en tant que routeur (*/proc/sys/net/ipv4/ip_forward*). Nous utilisons finalement un sous réseau. La solution aurait peut-être été amenée par iptables mais bon...

Pizza Box

Fichier */etc/sysconfig/network-scripts/ifcfg-eth0* sur la pizza box.

```
# Intel Corporation 80003ES2LAN Gigabit Ethernet Controller (Copper)
DEVICE=eth0
BOOTPROTO=none
BROADCAST=192.168.8.255
HWADDR=00:21:85:6a:40:4c
IPADDR=192.168.8.2
NETMASK=255.255.255.0
NETWORK=192.168.8.0
ONBOOT=yes
DNS1=192.168.1.3
DNS2=134.158.152.146
NM_CONTROLLED=no
TYPE=Ethernet
```

```
USERCTL=yes
PEERDNS=yes
IPV6INIT=no
DNS3=134.158.69.191
SEARCH=in2p3.fr
MTU=9000
```

3.4.4 Connectique

Définition IEEE802.3ab : 1000BASE-T

— Support minimum : câble en paires de cuivre torsadées non blindées de catégorie 5.
— Longueur maximale 100m

Cette définition est très importante. C'est elle qui permet d'utiliser le Gigabit Ethernet dans la majorité des installations actuelles.

Cela dit, les installations existantes auront certainement besoin d'une 'requalification'. Cette technologie utilise les câbles FTP (Foiled twisted pairs) de catégorie 5 au maximum de leur certification. De nouvelles catégories de câbles sont utilisables : 5enhanced à 100MHz, 6 à 200MHz, 6a à 500MHz et 7 à 600MHz. Il est recommandé de limiter au maximum les brassages intermédiaires dans les armoires de câblage.

Avec les cartes gigabit, il ne faut pas obligatoirement un câble Ethernet croisé pour tester la connection de Pc à Pc.

3.4.5 Tests des débits

Programmes utilisés

— Mesure via PING. En utilisant le script ci-dessous, on trouve des débits trés réduits.

```
#!/bin/sh
IP=192.168.2.1
NB_PING=10
SIZE=996   # +headers => paquets de 1024 octets

# le premier ping est toujours plus long (DNS...)
ping -c 1 $IP > /dev/null

# moyenne sur les pings suivant
averageTime=$(ping -s $SIZE -c $NB_PING $IP | tail -n 1 | cut -d'/' -f 5)

# On envoie 1024*8 bits en t milliseconde.
# d (b/s)  = 1024*8 * 1000/t
# d (Mb/s) = 1024*8 * 1000/t / 1024 / 1024
# d (Mb/s) = 8 * 1000/t / 1024

debit=$(bc << EOF
scale = 3;
8*1000/$averageTime/1024
EOF)
echo $debit Mb/s
```

— IPERF A télécharger puis compiler sur chaque plateforme.
Note : ne compile pas sur les pizza box. Il faut utiliser la version compiler sur le PC.
— Le programme suivant est analogue à IPERF. Il est utilisé en parallèle d'IPERF.

```
lpnhp90$ cvs -d :pserver:roche@lpnp90.in2p3.fr:/home/cvsroot co netSpeed
```

Permiers tests

Nous utilisons IPERF pour les tests et le programme maison pour affiner les résultats. Nous avons utilisé 3 câbles différents sans observer de différence de débits.

— Catégorie 6 UTP
— Catégorie 5e FTP
— Catégorie 5 UTP

Debits (bits/s) : TCP / UDP émission / UDP réception

depuis/vers	CES	PC NEC	PIZZA BOX
CES	320M / 470M / 440M	280M / 455M / 172M	280M / 460M / 173M
PC NEC	272M / 4.5G / 260M	9G / 10.5G / 15.2G	380M / 4.6G / 390M
PIZZA BOX	380M / 3.7G / 320M	941M / 3.4G / 670M	8.5G / 12.5G / 7.8G

Conclusions provisoires :

— On ne se rapproche de gigabit que dans la configuration PIZZA BOX vers PC NEC.
— Serait-ce possible que la puissance de la machine émettrice ait de l'influence ?
— Serait-ce possible que la puissance de la machine réceptrice soit aussi un facteur limitant ?
— La carte CES qui devra jouer le rôle de l'émetteur envoit au maximum **280** Mbits/s.

Tests après amélioration

L'amélioration effectuée est celle décrite dans le second mail de Ruppert donné ci-dessus :

— set prefetch size to maximum (`set bridge_lpci_prefetch 32`)
— use jumbo frames (`ifconfig eth0 mtu 9000`)

Source wikipedia : In computer networking, jumbo frames are Ethernet frames with more than 1,500 bytes of payload (MTU). Conventionally, jumbo frames can carry up to 9,000 bytes of payload, but variations exist and some care must be taken when using the term. Many, but not all, Gigabit Ethernet switches and Gigabit Ethernet network interface cards support jumbo frames, but all Fast Ethernet switches and Fast Ethernet network interface cards support only standard-sized frames.

Cette amélioration permet d'atteindre **520** Mbits/s en émission (et 688 Mbits/s en réception).

Optimisation de la taille des trames envoyées

A priori si l'on programme une socket TCP et que l'on envoie un buffer de la même taille que celui envoyé par IPERF on trouve le même débit.

TODO : Expliquer les faibles débits à la récéption avec le protocole UDP.

3.4.6 Test de la fibre optique

— Enfichement de la fibre optique.
Parfois, il faut faire pivoter les connecteur de 90 degrés afin de pouvoir les déconnecter.

Carte Processeur / Pc Nec

Les résultats ne sont pas dégradés.

Pizza Box vers Pc Nec

Le débit de l'ordre du gigabit par seconde n'est pas dégradés.
On se sert de cette connexion pour tester les nouvelles fibres.

3.4.7 Test de la configuration finale

— La carte processeur est relié au switch data via un cable ETHERNET.
— Le switch est configuré avec un MTU à 9000 et emprunte la liaison fibre optique
— Le convertisseur optique vers ETHERNET est relié à la pizza box par un cable ETHERNET

Cette configuration permet d'atteindre :
— **450** Mbits/s en émission TCP avec des envois de 10ko (soit 200 micro s)
— **550** Mbits/s en émission TCP avec des envois de 100ko
— **595** Mbits/s en émission TCP avec des envois de 500ko
— **599** Mbits/s en émission TCP avec des envois de 1Mo
— **595** Mbits/s en émission UDP avec des envois de 10ko (soit 160 micro s)

3.5 DMA

3.5.1 Introduction

Direct Memory Access. Accès direct à la mémoire. Méthode de transfert de données dans un ordinateur évitant d'avoir à utiliser le processeur et/ou une zone d'E/S standard. Le processeur lance donc le transfert DMA et peut continuer de faire des calculs indépendants (surtout grâce à sa mémoire cache), les périphériques ayant leur propre canal DMA évitent de faire la queue pour l'accès à la zone d'E/S. En fait, ceci est de moins en moins utilisé, car les processeurs sont désormais nettement plus rapides que le reste de la machine, ce qui n'était pas le cas quand le DMA a été mis au point.

Le rapport avec la choucroute :

The PCI-to-ISA bridge, frequently referred to as the South Bridge, connects the root PCI bus to the ISA (or EISA) bus. The South Bridge also typically incorporates the Interrupt Controller, IDE Controller, USB Host Controller and the DMA Controller.

Linux :

```
$ cat /proc/dma
```

3.5.2 User memory transfert

```
> For the moment I read cPCI data into a kernel buffer, do byte-swap
> then copy the kernel buffer into the user buffer, which is very time
> consuming.
>
> Is it easy to program the DMA to do the transfer directly from cPCI
> into user memory (the user then has to do byte-swap), that is when
> the blocks are greater than a memory page ?

It is not really easy, but the kernel provides an API for that.
You have to use get_user_pages(), PageReserved(), SetPageDirty() and
page_cache_release() do do that. Please have a look at the implementation of
XPC_DMA_EXEC in xpc_ioctl_dma() in xpc_dma.c for an example of how do do that.

Byte-swapping in should be fast, too. The PowerPC processor has special
load/store instruction for that which are just as fast as the non-swapped
variants. All you have to do is use some inline-asm.
```

3.6 Watch-dog

3.6.1 Introduction

cf le mail de Luisa :

```
To use the watchdog, one has to create a character special file:
# mknod /dev/watchdog c 10 130
Then the device has to be "pinged" regurarly by writing one or more characters to it.
This could be done by a C program, or a shell script.

A shell script would look like:

while true; do
        echo -n "x"
        sleep 20
done >/dev/watchdog

The  Linux kernel has not support for the watchdog of RIO3 and RIOC boards.
You get "no such device" error because as I said in the Linux kernel you currently have,
there is no driver for the watchdog of RIOC 4068/ RIOC 4065.
I have asked to our Linux developer to work at a watchdog driver for RIOC and RIO3 boards.
As soon as I have it I will send you.
```

3.6.2 test

```
# mknod /dev/watchdog c 10 130

# echo -n "x" > /dev/watchdog
bash: /dev/watchdog: No such device
```

Chapitre 4

Contrôleur Sécurité

Le Châssis sécurité est développé par le LAPP. Il est documenté à cette adresse : http ://lappweb.in2p3.fr/ panazol/web.php

Le LAPP prend de part son implication initiale dans le projet prend en charge les responsabilités suivantes :

— développement des cartes électroniques
— développement du driver
— développement d'un pilote

En revanche, le LAPP se dénonce son second engagement comme allant au delà de sa collaboration. Aussi le LAP ne fournira qu'à titre d'exemple d'utilisation un client/serveur doublé d'une modélisation des données en mémoire concrétisée par l'élaboration d'un fichier de configuration.

Suite au compromis oral opbtenu avec Mr Panazol, le LAPP continu à collaborer sur les parties suivantes :

— maintenance sur les cartes électroniques
— maintenance du driver
— maintenance du pilote
— maintenance de la modélisation des données

Il reste donc à développer :

— un serveur **Bunny** reprenant le plus possible les codes du LAPP.
— un controlleur CORBA pour ce serveur.

4.1 Démon Bunny

4.1.1 Introduction

Implémentation du serveur Bunny.

L'arborescence ci-dessous montre comment le code du LAPP (les feuilles) est appelé à partir de la mise en marche du serveur.

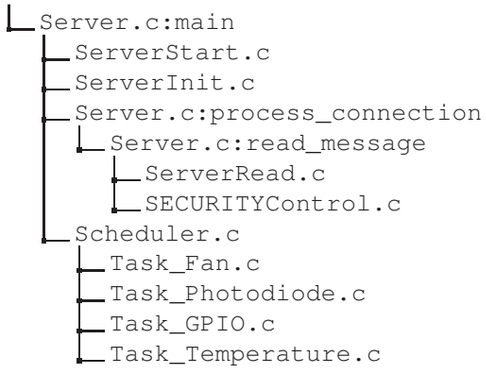


FIGURE 4.1 – *lancement du serveur* BUNNY

A l'initialisation, bunny met en route les ventilateurs de la façon suivante (selon le pilot) :

— `26 7 9f`
— `26 8 9f`
— `26 9 9f`
— `26 a 97`
— `26 0 ff`
— `26 1 ff`

4.1.2 Répertoires

— *Server* : Main
— *Control* : Analyseurs Grammaticaux
— *Scheduler* : Threads Panazol + Wiener
— *Util* : Misc + parser Panazol
— *Board* : ioctls Panazol

4.1.3 Comportement

Les 4 thread sont lancés périodiquement selon les fréquences inscrites dans les fichier *include*.

Le controleur corba est bloqué durant les connexion tant que celle-ci ne se lancent pas dans un thread. Ce comportement est hérité de Big.

4.1.4 Remontées des messages d'erreur

A priori les erreurs atteignent le contrôleur camera. Comment le contrôleur sécurité peut-il s'inscrire afin que Bunny lui envoit ses erreurs ?

```
Il faudrait que le contrôleur sécurité ait une fonction "CameraMessage" calculée sur
celle du CameraControleur.
Voir dans Camera2.idl:

    // Callback - Messages from the CPU
    void CameraMessage(in Dash::BlockAcceptor::Block data);
```

Ensuite, il faut envoyer à Bunny l'adresse corba. Dans CameraInterface_BigD.py:

```
    ##
    ## Send the controller node to the camera
    ## @param name CORBA address of Controller
    def SendControllerOp(self,name):
        self.Print("Send the controller node for %s"%(name))
        try:
            command = "BEGIN\nNODE CONTROLLER %s\nENDMSG"%(self.create_nodes_op(name,"CameraMessage"
            self.send_command(command)
        except Exception,x:
            traceback.print_exc()
            raise Camera2.CameraError("While contacting camera : %s"%str(x))
        return
```

4.2 Clonage d'un contrôleur CORBA

4.2.1 Introduction

Cette section donne la procédure suivie pas à pas pour cloner le module OWIS.

4.2.2 Clonage de l'interface CORBA

Il s'agit d'un fichier de spécification écrit en IDL. Afin de procéder pas à pas, nous ne détaillons pas cette étape et l'incluons dans la section suivante.

4.2.3 Clonage du serveur CORBA

Il s'agit d'un serveur écrit en C++.

— Duppliquer le module *owis* :

```
$ cd ~/cvs
$ cvs co owis
$ mkdir ~/src
$ cp -fr ~/cvs/owis ~/src/.
$ cd ~/src/owis
$ find . -type d -name CVS -exec rm -fr {} \;
```

— Modifier le fichier *Makefile* :

```
...
menage:
        @rmdir bin lib out stub Owis Owis__POA
```

— compiler :

```
$ make 2>/dev/null && echo "no soucis"
omniidl ... -Cstub idl/Owis.idl
g++ ... -c src/Controller.C -o out/Controller.o
g++ ... -c stub/OwisSK.cc -c -o out/OwisSK.o
g++ ... -o bin/Controller
g++ ... -c src/GPIBController.C -o out/GPIBController.o
g++ ... -o bin/GPIBController

$ make clean menage
$ make
```

— nouveau fichier C++ :

```
$ cp src/Controller.C src/ctrlSecurity.C
$ sed -i.sav -e 's|^\(PROGRAMS = .*\)|\1 ctrlSecurity|' Makefile
$ sed -i.sav -e 's|^\(SOURCES = .*\)|\1 ctrlSecurity|' Makefile
$ make
g++ ... -c src/ctrlSecurity.C -o out/ctrlSecurity.o
g++ ... -o bin/ctrlSecurity
```

— nouveau header C++ :

```
$ cp include/Controller.hh include/ctrlSecurity.hh
$ sed -i.sav -e 's|^#include "Controller.hh"|#include "ctrlSecurity.hh"|' src/ctrlSecurity.C
$ make
g++ ... -c src/ctrlSecurity.C -o out/ctrlSecurity.o
g++ ... -o bin/ctrlSecurity
```

— modification du *Makefile* :

```
MODULE = Owis

PROGRAMS = ctrlSecurity
SOURCES = ctrlSecurity
NONROOTPROGRAMS =
LINKROOT = on
```

```
LIB = none
#DEPS.owis = HSbase HSevent
#LIBS.GPIBController = gpib
USE_LIBS = dash monitor
#NODICT = 1
#INCLUDES = sash/HESSArray

include ${HESSROOT}/dash/Makefile.dash

menage:
        @rmdir bin  lib  out  stub Owis__POA Owis
```

— nouveau fichier IDL :

```
$ cp idl/Owis.idl idl/Security.idl
$ sed -i.sav -e 's|OWIS|SECURITY|g' include/ctrlSecurity.hh src/ctrlSecurity.C
$ sed -i.sav -e 's|Owis|Security|g'  include/ctrlSecurity.hh src/ctrlSecurity.C
$ sed -i.sav -e 's|^MODULE = Owis|MODULE = Security|' Makefile
$ make
```

— `erreur: monitor/ctrlSecurityEvent.hh : Aucun fichier ou répertoire de ce type`
Il faut commenter les lignes relatives à `ctrlSecurityEvent` dans *src/ctrlSecurity.C*.

— `erreur: 'POA_ctrlSecurity' has not been declared`

```
$ sed -i.sav -e 's|OWIS|SECURITY|g' idl/Security.idl
$ sed -i.sav -e 's|Owis|Security|g' idl/Security.idl
$ make
g++ ... -c src/ctrlSecurity.C -o out/ctrlSecurity.o
g++ ... -o bin/ctrlSecurity
```

— `erreur: invalid use of undefined type 'struct Monitor::SecurityEvent`
Il faut commenter les lignes contenant `fEvent->`.

4.2.4 Clonage du client non graphique CORBA

Il s'agit d'un client écrit en PYTHON.

— Duppliquer le fichier :

```
$ cd ~/src/owis/scripts
$ cp connect_controller.py clientSecurity.py
$ sed -i.sav -e 's|Owis|Security|g' clientSecurity.py
$ python clientSecurity.py
```

— `ImportError: No module named Security` Ajuster le chemin aux modules PYTHON générés via IDL

```
#sys.path.append(os.environ["HESSROOT"]+"/owis")
sys.path.append(os.environ["HOME"]+"/src/owis")
```

Contexte :

```
$ ./ctrlSecurity
...
Server: Bind name ctrlSecurity ...
...

$ nameclt list roche
ctrlSecurity

$ nameclt unbind roche/ctrlSecurity
$ nameclt remove_context roche
```

fichier *python/clientSecurity.py*

```
#!/usr/bin/python
import os
os.environ["LC_ALL"] = "C"
import sys
import thread
import time
sys.path.append(os.environ["HESSROOT"]+"/dash")
sys.path.append(os.environ["HESSROOT"]+"/dbgui")
sys.path.append(os.environ["HOME"]+"/src/owis")
from db import *

from omniORB import CORBA
import CosNaming
import Security,Security__POA
import Dash

global orb
orb = CORBA.ORB_init(sys.argv, CORBA.ORB_ID)
poa = orb.resolve_initial_references("RootPOA")
rootContext=orb.resolve_initial_references("NameService")._narrow(CosNaming.NamingContext)
poaManager = poa._get_the_POAManager()
poaManager.activate()

name = []
name.insert(0,CosNaming.NameComponent("ctrlSecurity", ""))
name.insert(0,CosNaming.NameComponent("roche", ""))
obj=rootContext.resolve(name)
controller = obj._narrow(Security.Controller)
if controller.GetState() == Dash.StateController.Safe:
    controller.Configure()

print "Position 1:",controller.GetPosition(1)
```

4.2.5 Clonage de l'interface graphique CORBA

Il s'agit d'un client graphique écrit en PYTHON.

— Duppliquer le fichier :
```
$ cd ~/src/owis/gui
$ cp GUI.py guiSecurity.py
$ sed -i.sav -e 's|Owis|Security|g' guiSecurity.py
$ python guiSecurity.py
```
— `ImportError: No module named Security` Ajuster le chemin aux modules PYTHON générés via IDL
```
#sys.path.append(os.environ["HESSROOT"]+"/owis")
sys.path.append(os.environ["HOME"]+"/src/owis")
```
— Nouveau nom CORBA :
```
$ nameclt list roche/Security
GUI
```

4.3 Contrôleur CORBA

4.3.1 Introduction

```
$ ./bin/Controller -n Security/Controller
                                                $ python gui/GUI.py
```

Envoie d'ordres

Les ordres transitent sous forme d'un pseudo codage XML semblable tout ou parti au contenu de fichier de configuration. Envoyer les ordres consiste à écrire ce texte dans une SOCKET.

Remontée des alarmes

Le client doit pouvoir s'inscrire au près du serveur afin de recevoir les alertes. A priori il s'agit du processus decrit ci-dessus mais inversé.

Remarque : notre client devient par conséquent également un serveur.

Récupération des valeurs

Afin de proposer un contrôle visuel, le client doit aussi être en mesure de récupérer les valeurs en vigueure sur le serveur. Celà consiste je suppose à envoyer un ordre puis à déchiffrer la chaîne reçue en retour.

4.3.2 Architecture du controleur Corba

Fichiers sources

```
ctrlSecurity
├── Makefile
├── idl
│   └── Security.idl
├── include .2 Controller.hh .1 src
│   └── Controller.C
├── scripts
│   ├── connect_controller.py
│   └── test_controller.py
└── gui
    └── GUI.py
```

FIGURE 4.2 – *fichiers source*

Interface CORBA

Le Makefile génère deux interface en C++ et en PYTHON :

```
$ omniidl -bcxx -Wba -I/usr/local/hess/dash/idl -Cstub idl/Security.idl
$ omniidl -bpython -I/usr/local/hess/dash/idl idl/Security.idl
```

```
  security
├─stub
│  ├─Security.hh
│  ├─SecuritySK.cc
│  └─SecurityDynSK.cc
├─Security_idl.py
├─Security
│  └─__init__.py
└─Security__POA
   └─__init__.py
```

FIGURE 4.3 – *fichiers de l'interface* CORBA

Serveur C++

Le Makefile génère deux objets et un executable *(cf cvs/dash/Makefile.dash)* :

```
g++ ... -c stub/SecuritySK.cc -o out/SecuritySK.o
g++ ... -c src/ctrlSecurity.C -o out/ctrlSecurity.o
g++ ... out/ctrlSecurity.o out/SecuritySK.o -o bin/ctrlSecurity
```

```
  security
├─out
│  ├─SecuritySK.o
│  └─Controller.o
└─bin
   └─Controller
```

FIGURE 4.4 – *fichiers du serveur* C++

Notez bien le paramètre !

```
$ ~cvs/security/bin/Controller -n Security/Controller
```

```
  roche
└─Security
   └─Controleur
```

FIGURE 4.5 – *noms* CORBA *du serveur* C++

```
$ nameclt list
roche/

$ nameclt list roche
Security/

$ nameclt list roche/Security
Controller
```

Clients PYTHON

Il y a redite puisque le python est un langage interprété.

```
security
├── scripts
│   ├── connect_controller.py
│   └── test_controller.py
├── gui
└── GUI.py
```

FIGURE 4.6 – *fichiers des clients* PYTHON

Simple connection

Ce script à la mérite d'être indépendant de la classe *Dash* dans le sens où il ne fait pas appel à ses fonctions pour initier la connexion au controleur CORBA.

fichier *cvs/security/scripts/connect_controller.py* :

```
...
from omniORB import CORBA
import CosNaming
import Security,Security__POA
import Dash

global orb
orb = CORBA.ORB_init(sys.argv, CORBA.ORB_ID)
poa = orb.resolve_initial_references("RootPOA")
rootContext=orb.resolve_initial_references("NameService")._narrow(CosNaming.NamingContext)
poaManager = poa._get_the_POAManager()
poaManager.activate()

name = []
name.insert(0,CosNaming.NameComponent("Controller", ""))
name.insert(0,CosNaming.NameComponent("Security", ""))
name.insert(0,CosNaming.NameComponent("roche", ""))
obj=rootContext.resolve(name)
controller = obj._narrow(Security.Controller)
if controller.GetState() == Dash.StateController.Safe:
    controller.Configure()

print "Position 1:",controller.GetPosition(1)
```

Interface non graphique

Ce script utilise les fonctions de la classe *Dash* pour établir la connection.

```
#!/usr/bin/env python

import os, sys

sys.path.append(os.environ["HESSUSER"])
sys.path.append(os.environ["HESSUSER"]+"/dash")
sys.path.append(os.environ["HOME"]+"/src/security")

from dash.python import StateController,Server,Message,LoopHandler
```

```
import Dash, Dash__POA
import Security, Security__POA

""" This is the Corba client"""
class SecurityUIClient(StateController.StateController_i):

    def __init__(self):

        StateController.StateController_i.__init__(
            self,
            name = "Security/UI",
            options = ["-ORBclientCallTimeOutPeriod","600000"]
            )

"""The main object gather Corba client and connected Corba server"""
class SecurityUI:

    def find_controller(self):
        """Client will try to find the controller """
        name = "Security/Controller"

        self.server = self.client.find_server(name)
        if self.server is not None:
            """Polymorphisme """
            self.server = self.server._narrow(Security.Controller)

    def __init__(self):

        self.client = SecurityUIClient()
        self.find_controller()

if __name__ == '__main__':
    os.environ["LC_ALL"] = "C"

    """ connect to the controller"""
    ui = SecurityUI()

    if ui.server == None:
        print "cannot contact Corba server"

    else:
        print "state :"
        ui.server.GetState()

    print ui.server

    print "Terminating (please ignore following insults)\n\t\t---\n"


  roche
└─Security
  └─UI
```

FIGURE 4.7 – *noms* CORBA *du client* PYTHON

```
$ nameclt list
roche/

$ nameclt list roche
Security/

$ nameclt list roche/Security
Controller
UI
```

Interface graphique

```
  roche
└─Security
  └─GUI
```

FIGURE 4.8 – *noms* CORBA *du* GUI PYTHON

```
$ nameclt list
roche/

$ nameclt list roche
Security/

$ nameclt list roche/Security
Controller
GUI
```

4.3.3 Implémentation du controleur Corba

Fonction ConfigureHardware

Chapitre 5

Slow Contrôle

5.1 Automates des démons

Ce document décrit les *runlevels* des serveurs Big, Zora, Emilie et Bunny.

Les actions ci-dessous seront exécuté préalablement sur chacun des châssis hébergeant un des quatre serveurs :

- Mise sous tension
- Chargement des modules Linux
  - Les drivers sont chargés au démarage par les scripts */etc/sysconfig/modules/*.modules*. L'extension ainsi que les droits d'execution sont r sont nécessaires.
  - L'emplacement des drivers chargés par `modprobe` est donné par les liens symboliques du répertoire */lib/modules/2.6.24.7-rt21-shl-3.2.3/hess2/*. Si l'on modifie ces liens, il faut alors ré-executer `# modprobe -a` afin de mettre à jour le fichier */lib/modules/2.6.24.7-rt21-shl-3.2.3/modules.dep*.
- Lancement du démon
  - Le démon est lancé de façon détachée d'un compte utilisateur par le script *rc.local*.
    ```
    ...
    /root/Bunny 2>/var/log/alertesBunny > /var/log/Bunny &
    ```
  - Le démon peut être relancé manuellement par le script */root/start*.sh*.
    ```
    #!/bin/bash
    nohup ./Bunny 2>/var/log/alertesBunny > /var/log/Bunny &
    ```

  Rq : On utilise `nohup` et les redirections (`>` et `2>`) pour détacher les exécutables de leurs terminaux. On choisi de se passer du démon SYSLOG par besoins de réactivité.

5.1.1 Automate générique

Remarque : Ces automates peuvent être désactivés via la ligne de commande ou via une socket :

```
# bunny -x

$ telnet camera17 1808
> BEGIN_
> RUNLEVEL GET EXPERT MODE
0: disabled
> RUNLEVEL ENABLE EXPERT MODE
> RUNLEVEL GET EXPERT MODE
1: enabled
> ENDMSG

# big

$ telnet camera12 1805
> BEGIN_
> RUNLEVEL GET EXPERT MODE
1: enable
> RUNLEVEL DISABLE EXPERT MODE
> RUNLEVEL GET EXPERT MODE
0: disabled
> ENDMSG
```

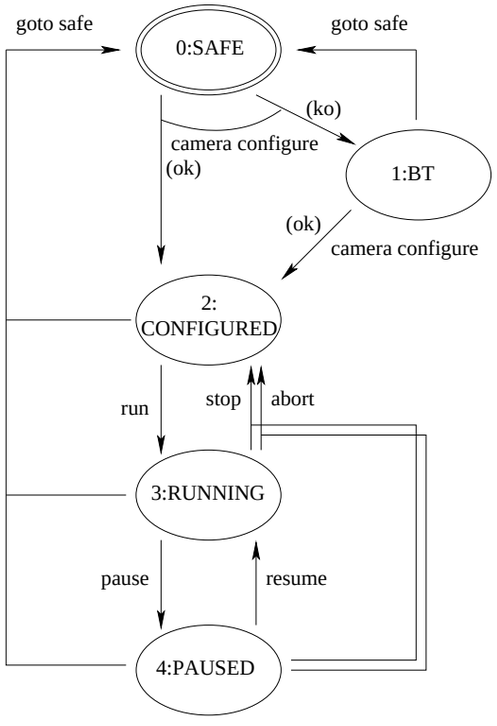


FIGURE 5.1 – AUTOMATE

Runlevel : Statut

— 0 : Safe (état initial et état poubelle).
— 1 : Basses tensions ON (état intermédiaire).
— 2 : Configured.
— 3 : Running.
— 5 : Paused

Transitions et évènements associés

transitions	évènement
*->0	***gotosafe***
0|1->1|2	***configure***
2->3	***start***
3->2	***stop***
3->2	***abort***
3->4	***pause***
4->3	***resume***

Les transitions ne sont **jamais** commandées en interne mais résulte d'ordres envoyés par un tiers, excepté pour la transition ***gotosafe*** faite à la fin de l'initialisation de chacun des serveurs.

— Bunny et les 4 automates de Big sont commandés par la ferme
— Emilie et Zora sont commandés par Big

la réponse de l'automate aux solicitations de changement d'état se fait de manière asynchrone

5.1.2 Bunny

Bunny doit contrôler 4 relais via 4 de ses sorties GPIO. Ces quatre relais bloquent ou laissent circuler d'autres GPIO en provenance de Big :

— les GPIO **10, 11 et 12** relayent la commande d'alimentation hautes tensions des tiroirs
— la GPIO **20** relaye la commande d'ouverture du capot

Il faut atteindre l'état "configuré" pour que les 4 relais deviennent passant. En revanche, ils re-basculent **automatiquement et définitivement** à l'état bloquant lorsque :

— l'alimentation des ventilateurs n'est pas stable
— la température excède un certain seuil
— les photo-diodes sont excitées au delà d'un certain seuil

Initialisation de Bunny

— L'alimentation des basse tension des ventilateurs est mise à ON
— Les ventilateurs sont lancés à vitesse constante (`0x9f` pour les petits et `0xff` pour les 2 gros extracteurs sur les côtés)
— Autoriser toutes GPIO excepté LID et HV
— La transition ***gotosafe*** est réalisée
— Lancement des threads énumérés ci-dessous

Threads de Bunny

— *Task_Wiener.c*
  — comparaison du courant et de la tension aux valeurs nominales
  — éventuelle inhibition des GPIO HV et LID
  — mise à jour du statut des basses tensions
— *Task_Fan.c*
  — lecture des la vitesse des ventilateurs
— *Task_Photodiode.c*
  — lecture des photo-diodes en entrée et calcul des seuils
  — éventuelle inhibition des GPIO HV et LID
  — mise à jour du statut des photo-diodes
— *Task_GPIO.c*
  — lecture des GPIO propres à Bunny
  — positionnement des sortie GPIO en fonction des GPIO lues en entrées *(désactivé)*
— *Task_Temperature.c*
  — lecture des capteurs de températures (moyenne sur n valeurs : 5) et calcul des seuils
  — ajustement de la vitesse des ventilateurs en fonction des seuils de température *(désactivé)*
  — positionnement des GPIO en fonction du maximum des seuils de température *(désactivé)*
  — éventuelle inhibition des GPIO HV et LID
  — mise à jour du statut des températures

Automate de Bunny

Évènement	tâches pré-requises au changement d'état
gotosafe	Bloquer les GPIO LID et HV *(fait)*
configure	Tester l'état non erroné des basses tensions *(fait)* Tester l'état non erroné des températures *(fait)* Tester l'état non excité des diodes-pin *(fait)* Autoriser les GPIO LID et HV *(fait)*
start	
stop	
abort	
pause	
resume	

exemple :

```
telnet camera17 1808
> BEGIN_
> RUNLEVEL TASK LIST
> RUNLEVEL GOTOSAFE
> RUNLEVEL CONFIGURE
> RUNLEVEL GET RUNLEVEL (questionnement par Big)
> RUNLEVEL GET STATUS   (questionnement par Big)
> ENDMSG
```

5.1.3 Big

Le serveur Big assume les tâches suivantes :

— Multi-FIFO. **(je ne comprend pas)**
— Monitoring.
— Ordres de haut niveau.

Initialisation de Big

— transition ***gotosafe*** de l'automate principal
— Lancement des threads énumérés ci-dessous

Threads de Big

— *Task_Camera_Status.c* : envoie de données de monitoring à la ferme
— *Task_Drawer_Monitoring.c* : envoie de données de monitoring à la ferme
— *Task_Wiener.c* : comparaison du courant et de la tension aux valeurs nominales. En cas d'erreur, envoyer un warning à la ferme *(à faire)*

Automate principal de Big

Évènement	tâches pré-requises au changement d'état
gotosafe	Transition ***gotosafe*** de l'automate du Emilie *(a faire)* Transition ***gotosafe*** de l'automate du Zora *(a faire)* Transition ***gotosafe*** de l'automate du capot *(fait)* Transition ***gotosafe*** de l'automate des triggers *(fait)* Transition ***gotosafe*** de l'automate des hautes tension *(fait)* Désactivation du thread Drawer_Monitoring *(à faire)* Basses tension mise à OFF *(fait)* Arrêt de l'envoi des données de monitoring à la ferme *(à faire)*
configure	Vérifier la température des DAQ via Bunny (runlevel >= 1 & statut correct) *(fait)* Basses tension mise à ON si OFF *(fait)* Broadcast sur SLC ready *(à faire)* Broadcast sur DAQ ready *(à faire)* Vérifier la cohérence des données envoyées par la ferme *(à faire)* Tester la réponse de Zora aux trigers soft *(à faire)* Tester la réponse d'Emilie *(à faire)* Paramétrage SAM *(à faire)* Envoi des paramètres SLC, DAQ et SAM à Emilie et Zora *(à faire)* Envoie de l'évènement ***configure*** à Emilie et Zora *(à faire)* Activation du thread Drawer_Monitoring *(à faire)* **DRAWER FF SET_CNTRL_SLC**
start	Vérifier les données envoyées par le "star controller" *(à faire)* Paramétrage NODES *(à faire)* Envoi des paramètres des NODES à Emilie et Zora *(à faire)* Envoi d'un "run header" aux "data receiver" *(à faire)* Envoie de l'évènement ***start*** à Emilie et Zora *(à faire)*
stop	Envoie de l'évènement ***stop*** à Emilie et Zora *(à faire)* Sans réponse d'Emilie ou de Zora, Big s'auto envoi ***abort*** *(à faire)* Envoi d'un "run tailer" aux "data receiver" *(à faire)*
abort	Tue puis relance Emilie et Zora via un autre serveur embarqué sur leur carte *(à faire)*
pause	Envoie de l'évènement ***pause*** à Emilie et Zora *(à faire)*
resume	Envoie de l'évènement ***resume*** à Emilie et Zora *(à faire)*
kill	Sortie du programme *(à faire)*

rq : Big possède la transition supplémentaire *kill*.

exemple :

```
telnet camera16 1805
> BEGIN_
> RUNLEVEL MAIN GOTOSAFE
> RUNLEVEL MAIN CONFIGURE
> RUNLEVEL TASK LIST
> ENDMSG
```

5.1.4 Zora *(à faire)*

Le serveur Zora assume les tâches suivantes :

— Lecture des données depuis le bus DMA.
— Timestamps via le **'local module'**.
— Envoi des données à la ferme.

Ce serveur est piloté via un sous automate de Big.

Initialisation de Zora

— ?

Automate de Zora

Évènement	tâches pré-requises au changement d'état
gotosafe	
configure	**DRAWER FF SET_CNTRL_SAM_REGISTER** **DRAWER FF SET_CNTRL_SAM_ND** **DRAWER FF SET_CNTRL_DAQ**
start	Envoi des données à la ferme *(à faire)*
stop	Arrêt de l'envoi des données à la ferme *(à faire)*
abort	
pause	Arrêt de l'envoi des données à la ferme *(à faire)*
resume	Envoi des données à la ferme *(à faire)*

5.1.5 Emilie *(à faire)*

Le serveur Emilie assume les tâches suivantes :
— Lecture des cartes triggers.
— Choix d'une stratégie de triggers (Soft, Interne ou Externe)
— Timestamps via le **'local module'**.
Ce serveur est piloté via un sous automate de Big.

Initialisation d'Emilie

— ?

Automate d'Emilie

Évènement	tâches pré-requises au changement d'état
gotosafe	
configure	**DRAWER FF SET_CNTRL_TRIGGER** **DRAWER FF SET_CNTRL_L2**
start	Envoi des données à la ferme *(à faire)*
stop	Arrêt de l'envoi des données à la ferme *(à faire)*
abort	
pause	Arrêt de l'envoi des données à la ferme *(à faire)*
resume	Envoi des données à la ferme *(à faire)*

5.1.6 Automate des hautes tensions

Évènement	tâches pré-requises au changement d'état
gotosafe	Hautes tension mise à OFF *(à faire)*
configure	Hautes tension mise à ON *(à faire)*
start	**DRAWER FF SET_VOLTAGE_VALUE**
stop	
abort	
pause	
resume	

exemple :

```
telnet camera16 1805
> BEGIN_
```

```
> RUNLEVEL HV GOTOSAFE
> ENDMSG
```

5.1.7 Automate des triggers

Évènement	tâches pré-requises au changement d'état
gotosafe	Désactivation des triggers *(à faire)*
configure	Activation des triggers *(à faire)*
start	
stop	
abort	
pause	
resume	

exemple :

```
telnet camera16 1805
> BEGIN_
> RUNLEVEL TRG GOTOSAFE
> ENDMSG
```

5.1.8 Automate du capot

Évènement	tâches pré-requises au changement d'état
gotosafe	Fermeture du capot *(à faire)*
configure	Ouverture du capot *(à faire)*
start	
stop	
abort	
pause	
resume	

exemple :

```
telnet camera16 1805
> BEGIN_
> RUNLEVEL LID GOTOSAFE
> ENDMSG
```

5.1.9 Todo list

Gestion des serveurs

- Créer un watchdog soft sur la carte slow contrôle qui lance Big si ce dernier n'est pas en cours d'execuction.
- Créer 2 démons capables de tuer puis relancer Emilie et Zora sur commande de Big (évènement abort)

Remonté des erreur vers l'opérateur

Utiliser les primitives suivantes pour afficher les messages. On pourra facilement éliminer ces sortie via des `#define`.

```
int STATUSMessage(char *message);
int DBUGMessage(char *message);
int INFOMessage(unsigned short int errorid,char *message,...)
int CAUTIMessage(unsigned short int errorid,char *message,...)
```

```
int WARNMessage(unsigned short int errorid,char *message,...)
int ERRORMessage(unsigned short int errorid,char *message,...)
```

— Gestion des erreurs atteignant le controleur caméra :
  — INFO
  — CAUTION
  — WARNING
  — ERROR bloque le controleur caméra.
— Les codes de retour (un code par erreur) sont définit dans le fichier *Driver/include/Message.h*
— Le statut est renvoyé à chaque solicitation des automates.

Basses tensions

Actions à effectuer en conséquences aux instabilités :
— Bunny : positionnement d'un flag interrogeable par Big (fait)
— Big : envoi d'un warning à la ferme

Timestamps via le 'local module'

Le '**local module**' est développé par le MPIK.

Le '**local module**' permet de dater les évènements et d'identifier les destinataires à qui les serveurs devront les envoyer ; ie le '**local module**' identifie chaque évènement via les trois champs suivants :
— event
— brunch
— IP
Les serveurs ZORA et EMILIE doivent communiquer avec le '**local module**'. **Question :** comment communiquent-ils (schéma énumérant les flux) ?
Plugger la carte et demander la doc à Patrick

Big

Définir une fréquence ainsi que la politique de priorité du thread de supervision.
— toutes les 100 ms
— prioritaire sur les ordres
Définir également un quotient de cette fréquence pour chacune des activité du superviseur :
— rapide : Ré-allumage des pixels via le **'star controller'**
— ? : supervision des basses tensions

Températures des DAQ

Prévu dans le serveur Big :
— un seuil de déclenchement de l'alerte
— un seuil (plus faible) d'annulation de l'alerte

Hautes tensions

La haute tension nominale est de 1000 Volts.
On peut avoir les hautes tension en marche (moins de 400 Volts) et le capot ouvert lors des "changement de priorité".

Capot

— VMC IO à installer.
— Contrôle depuis le serveur de console blackbox ?

5.2 Star Contrôleur

5.2.1 Introduction

Ce document décrit la gestion des pixels de la caméra.
Les pixels peuvent être éteind

— suite à la détection d'un voltage trop élevé
— ou en prévention.

Il faut par ailleurs rallumer les pixels éteints.

5.2.2 Code HESS1

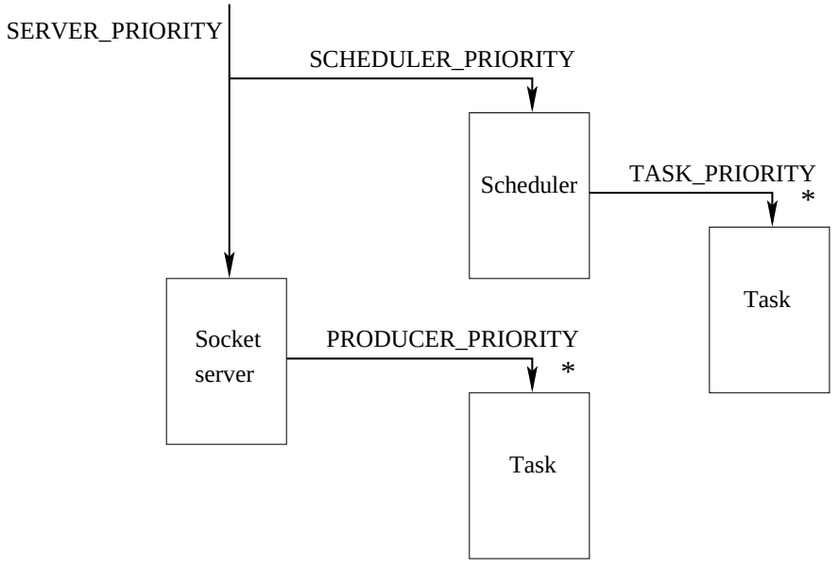


FIGURE 5.2 – Schéma des threads des serveurs Hess2

Dans HESS1, Le code utilitaire de gestion des pixels est appellé au niveau du scheduleur du serveur de socket et de l'acquisition.

— *Sdaq/Server/Daqd/VERSION_0.00/src/acq_camera.c* :
This process is real time acquisition of the HESS telescope camera on interrupt from the IO register board

```
void *thread_inbuf (void *arg) {
 /* block all signals */
 while(1) {
  sigwait(&ensemble, &num);
  if(sig_monitoring == SIG_CAMERA_HTMON) {
   headp = (Event_Header_t *) &cbuff[inbuff];
   Do_Slc ((Event_Header_t *) &cbuff[inbuff],...)

   //get present time
   if(!DrawerConfig) {
    drawer_ptr = (unsigned short *) &cbuff[inbuff] + header_size;
    while(i < headp->data_size)
     DrawerSurvey(drawer_ptr + i, papat_desc, 0xf, 0xf, 0x3b, time_of_endread.tv_sec);
}}}}

int main(int argc, char *argv[])
{
 tcam_desc = open("/dev/camera", O_RDWR);
 common_def.des_camera = tcam_desc; // pass camera-device desc

 status = pthread_create(&thr_inbuf, &thread_attr, thread_inbuf, NULL);
 ...
 status = pthread_join(thr_inbuf, &thread_result);
 exit(0);
}
```

— *Sdaq/Server/Slcd/VERSION_3.00/Consumer/Task_DATA_Consumer.c* :
Scan FIFO_Buffer and search for slow control or DAQ event Check header and tailer of the block, block identificator and corresponding size.
Store data in DATA_Buffer to be read end send by other consumers

```
void Task_DATA_Consumer(void *mytask){...
do{...
 /* Store locale GPS/CPU time stamp */
```

```
 locate_time_sec = *(int *)(pDATA+10);

 /* check online star monitoring prediction if PM are not in transition state
 SurveyStars(locate_time_sec, slcd.pattern);

 /* Loop on all 4xFIFO */
 for(ii=0; ii<4; ii++)
  /* If high voltage is on then check the behaviour of photo multipliers */
  err = DrawerSurvey(pDATA_save,...)

} while (task->_status & TASK_STATUS_CREATED);
pthread_exit(0);
}
```

— *Sdaq/Server/Slcd/VERSION_3.00/Util/Warning.c*

```
int SwitchOffPixel(int drawer, int pixel, int fifo, int pattern, char *reason){
 Config[drawer].cntrl_ht_16.HV_status &= ~(0x1<<pixel);
 ierr = cPCI_FIFO_Write_Block(fifo, (short *)&Config[drawer].cntrl_ht_16, pattern);
}

int SwitchOnPixel(int drawer, int pixel, int fifo, int pattern, char *reason){
 Config[drawer].cntrl_ht_16.HV_status |= 0x1<<pixel;
 ierr = cPCI_FIFO_Write_Block(fifo, (short *)&Config[drawer].cntrl_ht_16, pattern);
}

int SurveyStars(unsigned long long int time, unsigned int pattern){
 while(1){
  if (!ioctl(common_def.des_camera, SURVEY_STAR_CHECK, &star)) break;
  drawer_num = DrawerId2Number(star.drawer);

  switch(star.InOrOut){
  case 1: " Star enter (drawer 0x%X : HT=0x%X)"
  SwitchOffPixel(...)
  case 0: " Star leave (drawer 0x%X : HT=0x%X)"
  SwitchOffPixel(...)
  }

  ioctl(common_def.des_camera, SURVEY_STAR_REMOVE, star.id);
}}

int DrawerSurvey(short *pDATA,...){
 /* Check limit on HTI value */
 SwitchOffPixel(drawer_num, pixel, fifo, pattern, message);
 /* Check HT value send for this drawer */
 SwitchOffPixel(drawer_num, pixel, fifo, pattern, message);
 /* Check noise intensity - HTI vs HTV */
 SwitchOffPixel(drawer_num, pixel, fifo, pattern, message);
}
```

— *Sdaq/Server/Slcd/VERSION_3.00/src/SURVEYControl.c* :
Control sun, moon and star

```
int SURVEYControl(char *Buffer){
 if (!Look4Key(local_buffer,"SUN",3))...
 else if (!Look4Key(local_buffer,"MOON",4))...
 else if (!Look4Key(local_buffer,"STAR",4))...
 else if (!Look4Key(local_buffer, "CLEAR", 5))...
 else if (!Look4Key(local_buffer,"DUMP",4))...
}
```

— *Sdaq/Driver/tcamera/drv_camera.c* :
Linux driver for the HESS telescope camera this is a logical driver, no hardware is directly involved

```
    case SURVEY_SET_SUN:       Run.survey.sun  = *(Solar_t *)ioctl_param;        break;
    case SURVEY_SET_MOON:      Run.survey.moon = *(Solar_t *)ioctl_param;        break;
    case SURVEY_STAR_ADD:
      if (!(star = kmalloc(sizeof(struct star_t), GFP_KERNEL))) return(ENOMEM);
      *star = *(Star_t *)ioctl_param;
      /* gestion de liste ... */
      Run.survey.nbStars++;
      return(star->id);
    case SURVEY_STAR_REMOVE:
    case SURVEY_STAR_CHECK:
    case SURVEY_GET_STAR_ID:
    case SURVEY_GET_STAR_NUMBER:
    case SURVEY_GET_SUN:
    case SURVEY_GET_MOON:
    case SURVEY_GET_NUMBER_OF_STARS:
```

Question : Où se trouve le code du star controller ?

5.2.3 Extinction préventive

Grammaire donnée par le controler :

```
/* star survey */
  else if (!Look4Key(local_buffer,"STAR",4))
    {
      local_buffer +=5;
      if (sscanf(local_buffer,"IN %2c%02d %Ld",drawer,&pixel,&val) == 3)
      ...
      else if (sscanf(local_buffer,"OUT %2c%02d %Ld",drawer,&pixel,&val) == 3)
```

On simulera le Star Controller par un fichier à l'aide de la commande date.
Question : s'agit-il d'autre chose que du controleur ?

5.2.4 Voltage trop élevé

```
int DrawerSurvey(short *pDATA,...)
{
   ...
   SwitchOffPixel(...);
}
```

Question : à implémenter dans *Task_Drawer_Monitoring.c* ?

5.2.5 Ré-allumage des pixels

Il faut élaborer la liste des pixels éteints.
Tableau des pixels :

```
typedef struct
{
  ...
  Drawer_HV_16_t           cntrl_ht_16;
  ...
} Slc_config_16;

SEXTERN Slc_config_16 Config[__NUMBER_OF_DRAWERS];
```

Réaction :
— 4 à 6 pixels étends : étoiles (pas grave)
— 6 à 120 pixels étends : étoile filante (pas grave)
— > 120 : voiture ou éclairs (on stop l'acquisition)
question : à implémenter dans une nouvelle tâche du squeduler ?

Chapitre 6

Accéder aux cartes processeurs

6.0.6 Introduction

Nous utilisons un serveur de port série (192.168.1.62) :
— pour se connecter aux cartes controleur
— pour communiquer avec le GPS
— éventuellement pour actionner des relais
Voici la doc du BLACK BOX Secure Device Server 16-Port (LEB410A-16) utilisé :
— guide utilisateur
— interfaces shell

Configuration via le serveur web et mot le passe habituel. En cas de perte d'adresse Ip, on peut -en théorie- en affecter une temporairement via le serveur ARP du PC client (attendre 30 secondes) :

```
# arp -s 192.168.1.61 00:80:d4:04:42:A8
# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.1.61             ether   00:80:d4:04:42:a8   CM                    eth1
```

6.0.7 Connexion aux cartes controleurs

192.168.1.62 :2002	camera12
192.168.1.62 :2003	camera13
192.168.1.62 :2006	camera16
192.168.1.62 :2007	camera17

Exemple de connexion :

```
$ telnet 192.168.1.62 2002
login:
...
(fermer la fenêtre)

$ ssh -p 2502 root@192.168.1.62
root@192.168.1.62's password:
Sugar Hat Linux Toolbox for CES boards release 3.2.1
login:
...

$ telnet 192.168.1.62 2002
telnet: connect to address 192.168.1.62: Connection refused
```

Attention : une seule connexion à la fois à chaque carte.
On peut controller et annuler les connexions en cours via l'interface web.

Chapitre 7

Switchs embarqués

7.0.8 Introduction

Nous utilisons 3 switchs BLACK BOX :
— **LGB2008A** : switch 8 ports RJ45 / 2 ports optiques => *switch data*
— **LGB2008A** : switch 8 ports RJ45 / 2 ports optiques => *switch trigger*
— **LGB2002A-R2** : switch 16 ports RJ45 / 4 ports optiques => *switch control*
Un *Connecteur* est enfiché dans chacun des switch :
— **LGB200C-MLC** : 1000Mbps SFP 850nm Multi-mode
A l'autre bout de la fibre optique, des *Media Converter* réalise la conversion cuivre/optique :
— **LMCS212AE-SC** : 1000 Base-T to 1000 Base-SX/LX
Voici la doc rassemblée :
— switch 16 ports **LGB2002A-R2**
Voici le numéro de hotline de BLACK BOX qui équipe donc tout le réseau de la caméra : **08 20 07 09 11** ou **01 45 60 67 17**.

7.0.9 Connexion aux switchs

Configuration via le serveur web (mot de passe habituel)
— switch #1 (8 ports) : http ://192.168.7.51/
Penser à passer le MMTU à 9000 sur le port utilisé par zora.
— switch #2 (8 ports) : http ://192.168.1.52/
— switch 16 ports : http ://192.168.1.53/
... ça peut aider (la pizza box fige son bureau lorsqu'elle perd le réseau) :

```
# ifdown eth0
# ifconfig eth0 192.168.1.2
# route add -net 192.168.1.0 netmask 255.255.255.254 dev eth0
```

Deuxième partie

Reprise de l’existant

Chapitre 1

Savoir Faire

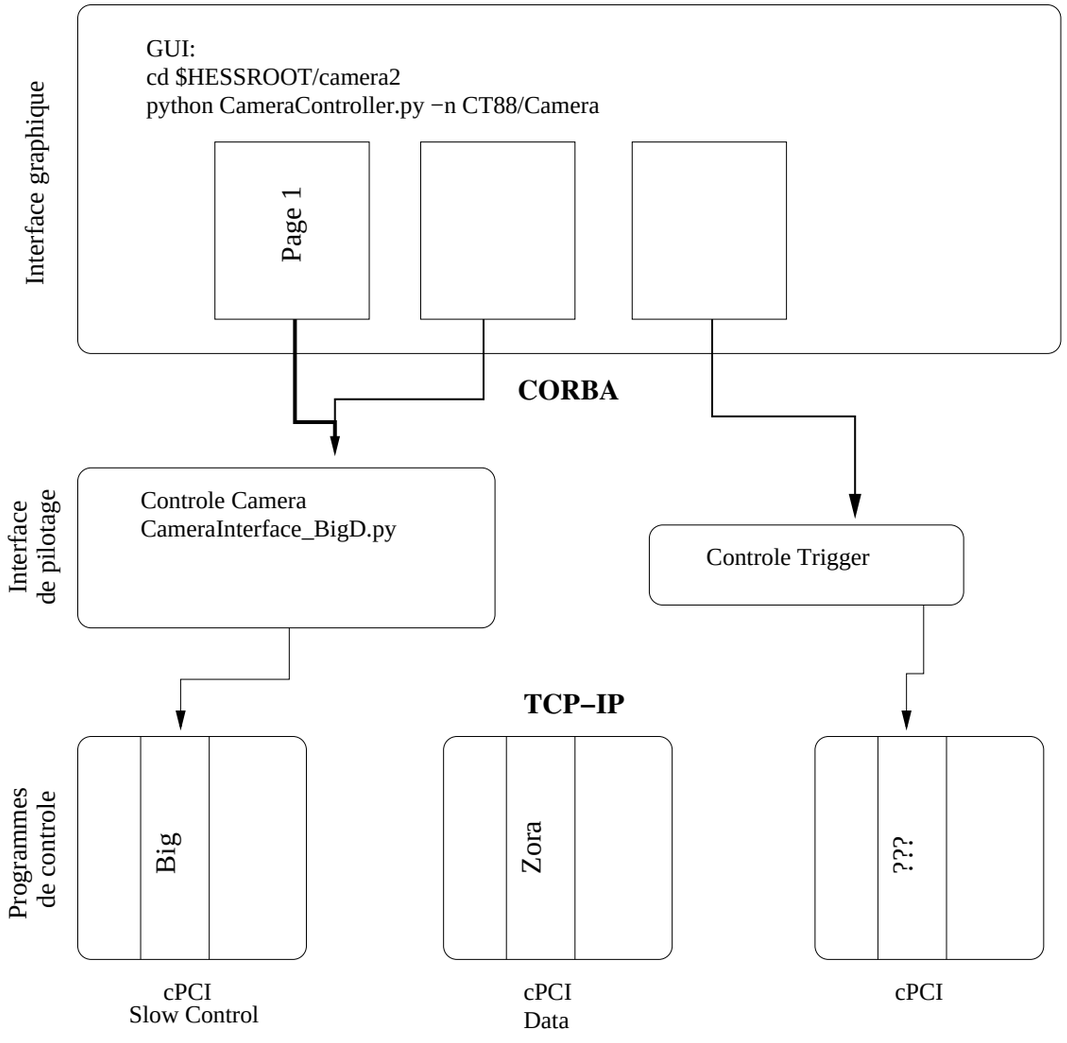


FIGURE 1.1 – Schéma d'organisation du contrôle caméra HESS-II

— **Programme de contrôle**
Un programme de contrôle installé sur chacun des châssis d'acquisition. Ce programme est spécifique à chaque châssis. Il contrôle les cartes d'acquisition correspondante via les drivers, gère leur configuration et gère également l'envoi des données à la ferme de calcul.
Ce programme comporte un interpréteur de commande, écoutant sur un port spécifique et permettant de recevoir la configuration dans un format ASCII simple. De façon générale, la configuration des cartes électroniques se fait au travers de 'blocs de contrôle', au format binaire, envoyés par le programme de contrôle sur les FIFO ou à travers le bus PCI et décodé par les FPGA installés sur les cartes d'électronique ou sur les tiroirs.
Le travail de configuration consiste donc pour l'essentiel, à décoder les commandes de configuration et à coder les blocs de contrôle. En outre, le programme de contrôle gère la cohérence de la configuration caméra.
— **Interface de pilotage**
Une interface de pilotage, qui est un petit module PYTHON, sans interface graphique, dont le rôle est de gérer la communication avec les programmes de pilotage sur CPCI. Ce programme existe actuellement pour la partie **BigD** (slow control caméra).

L'indépendance avec l'interface graphique permet de garantir le fonctionnement même en cas de plantage de l'interface graphique. Cette dernière est alors relancée et se reconnecte automatiquement.

— **Interface graphique**
Une interface graphique proprement dite, reflétant graphiquement la configuration de l'électronique. Dans un premier temps, il est souhaitable d'avoir un miroir de l'électronique, c'est à dire une page de configuration par carte électronique, afin de permettre aux ingénieurs électroniciens d'accéder à chaque élément de la configuration. Dans un second temps, l'interface graphique sera connectée à l'acquisition centrale et devra donc gérer la configuration automatique de la caméra selon des stratégies de prises de données.

1.1 Corba

1.1.1 Introduction

L'architecture est déjà mise en place via un client/serveur générique.

1.1.2 Modèle client/serveur générique

Il faut comprendre que le controleur, tout comme son interface graphique sont à la fois client et serveur.
- Serveur spécifique
  - serveur de SOCKET par exemple
- Contrôleur
  - client SOCKET par exemple
  - serveur CORBA
- GUI du contrôleur
  - client CORBA
  - serveur CORBA
- GUI caméra
  - client CORBA

1.1.3 Interfaces Dash : :Controller

```
$ echo $HESSUSER
/home/roche/hess

$ cd ~/hess
$ cvs co dash
$ cd dash && make
```

L'interface spécifie l'objet vu par le serveur.
Cf fichier */hess/dash/idl/Dash.idl*
Les méthodes disponible pour la classe **Controller** sont implémentées ici :
- */hess/dash/python/Server.py* Cette classe implémente les mécanismes de base propre à chaque serveur CORBA.
  - Le constructeur contacte les servers POA (Portable Object Adapter).

```
print "Initializing ORB"
orb = CORBA.ORB_init(options, CORBA.ORB_ID)
poa = orb.resolve_initial_references("RootPOA")
rootContext=orb.resolve_initial_references("NameService")._narrow(CosNaming.NamingContext)
poaManager = poa._get_the_POAManager()
poaManager.activate()
thread.start_new_thread(startorb,())
```

  - `find_server(self,name)` : Find a server with a given name, and try to contact it

  Voici les méthodes publiques :
  - `Delete` `// Terminate the server process`
  - `Ping` `// Check if the connection is up`
  - `Initialized` `// Check if the process is initialized ; To be overloaded`

```
Server
└─Message
  └─StateController
    └─Controller
```

FIGURE 1.2 – héritages

— */hess/dash/python/Message.py*
Cette classe ajoute une couche permettent le débugage équivalente au démon SYSLOG.
Voici les méthodes publiques :
— AddMessageReceiver
— RemoveMessageReceiver
— TakeMessage
— Take

— */hess/dash/python/StateController.py*
Cette classe fait de chaque controleur une machine à état.
Par ailleurs, cet objet fait intervenir la notion de thread. Les commandes sont alors suspendues/annulées... par ce biais.
Voici les méthodes publiques :
— GetState
— GetProcessStatus
— SetProcessStatus
— GetTransition
— IsInTransition
— Action
— TakeRunParameters
— GetRunParameters
— GotoSafe
— Configure
— Start
— Stop
— Resume
— Interrupt
— IniGotoSafe
— IniConfigure
— IniStart
— IniStop
— IniPause
— IniResume

— **Controller**
Cette classe virtuelle pure, en plus des classes décrites ci-dessus, hérite d'autres classes mais qui ne sont pas implémentées.
Nos objets implémenterons cette classe.

remarque : En python les méthodes virtuelles pures peuvent ne pas être implémentées.

1.1.4 Utilisation de Dash

Nous intégrons l'architecure générique via la classe **Controller** :

— Dans notre interface IDL

```
#include "Dash.idl"

module Onlinecalibrator
{
  interface Controller: Dash::Controller
```

— Dans notre serveur, la place de la première classe héritée étant imposée (sinon le client ne se connecte pas), on en déduit que c'est elle qui joue le rôle déterminant.

```
from dash.python import StateController

class OnlinecalibratorController_i(Onlinecalibrator__POA.Controller,
                                   StateController.StateController_i):
```

```
        def __init__(self, name, ledServer, ledPort):
            StateController.StateController_i.__init__(self,name,options=["-ORBclientCallTimeOutPer
```

— Dans notre client on retrouve également l'implémentation de la classe `StateController`. La méthode `find_server` de l'instance locale (`self.controller`) permet d'initialiser l'instance distante (`self.server`). Dès lors l'instance locale ne sert plus.

```
from dash.python import StateController,Server,Message
import Dash, Dash__POA

class OnlinecalibratorUIController(StateController.StateController_i):

    def __init__(self,
                 name = "Onlinecalibrator/GUI",
                 options = ["-ORBclientCallTimeOutPeriod","600000"]):

        StateController.StateController_i.__init__(self,name = name,options = options)


class OnlinecalibratorUI:

    def __init__(self,name = "Onlinecalibrator/GUI"):

        self.controller = OnlinecalibratorUIController(name = name)
        self.find_controller()

    def find_controller(self):
        """Try to find the controller """
        name = "Onlinecalibrator/Controller"

        self.server = self.controller.find_server(name)
        if self.server is not None:
            self.server = self.server._narrow(Onlinecalibrator.Controller)
```

Un exemple d'utilisation est donné dans la section contrôleur

1.2 Interface de pilotage

1.2.1 Introduction

Il s'agit de développer une interface cliente dialoguant avec un serveur spécifique. Cette interface devra également jouer le rôle de serveur via l'interface CORBA.

1.2.2 Interface IDL

Le nouveau controlleur hérite de la classe *Dash : :Controller*.

```
#include "Dash.idl"

module NouveauModule
{
  enum ErrorType { Timeout, CmdError, ConnectionError, EmergencyStop };

  exception OnlinecalibratorError {
    ErrorType type;
    string message;
  };

  interface Controller: Dash::Controller
  {
    void nouvelleFonction(in short value) raises (OnlinecalibratorError);
    ...
  };
};
```

1.2.3 Contrôleur

Par contrôleur on entend :
— un client d'une appliquation.
— un serveur CORBA

```
#!/usr/bin/python
import os,sys
import string
import time

# imports generic
import Dash, Dash__POA
from dash.python import StateController

# imports specific
import NouveauModule, NouveauModule__POA
import tcpClient

# Attention: la place du 1er héritier n'est pas annodine !!
class NouveauModuleController_i(NouveauModule__POA.Controller,
                                StateController.StateController_i)
                                ...):

    def __init__(self, name, ...):
        StateController.StateController_i.__init__(self,name,
                            options=["-ORBclientCallTimeOutPeriod","600000"])
        ...
```

```
        self.stop = 0

    def MainLoop(self):
        while not self.stop:
            time.sleep(0.1)

    def nouvelleFonction(self, value):
        ...mon code de connexion au serveur C...


if __name__ == '__main__':

    # naming service's name given by `nameclt list LOGIN`
    name = "NouveauModule/Controller"

    c = NouveauModuleController_i(name, ...)
    c.MainLoop()
```

1.2.4 Client pour les tests

Il s'agit juste d'un client CORBA. Le serveur CORBA étant lui le contrôleur décrit juste au dessus.
Par client on entend :
— un client CORBA

```
#!/usr/bin/env python

import os, sys
from dash.python import StateController,Server,Message,LoopHandler
import Dash, Dash__POA
import NouveauModule, NouveauModule__POA


class NouveauModuleUIController(StateController.StateController_i):

    def __init__(self,
                 name = "NouveauModule/GUI",
                 options = ["-ORBclientCallTimeOutPeriod","600000"]):

        StateController.StateController_i.__init__(self,name = name,
                                                   options = options)


class NouveauModuleUI:

    def __init__(self,name = "NouveauModule/GUI"):

        self.controller = NouveauModuleUIController(name = name)
        self.find_controller()

    def find_controller(self):
        """Try to find the controller """
        name = "NouveauModule/Controller"

        self.server = self.controller.find_server(name)
```

```
        if self.server is not None:
            self.server = self.server._narrow(NouveauModule.Controller)


if __name__ == '__main__':

    ui = NouveauModuleUI()

    if ui.server == None:
        print "cannot contact Corba server"

    else:
        ui.server.nouvelleFonction(0xab)

    print "Terminating"
    ui.controller.destruct()
```

1.2.5 Makefile

```
.PHONY: all clean

all:
        omniidl -bpython -I/usr/local/hess/dash/idl idl/NouveauModule.idl

clean:
        @rm -fr NouveauModule
        @rm -fr NouveauModule_idl.py
        @rm -fr NouveauModule__POA
```

1.2.6 Graphisme

Client graphique

Par client graphique on entend :
— un client CORBA
— une interface graphique
— un serveur CORBA

Attention, il s'agit d'un client CORBA capable de joindre le controlleur (au même titre que le client décrit ci-dessus). Cependant ce client est par ailleurs lui même un serveur CORBA vis à vis de l'interface graphique de plus haut niveau (l'interface *caméra contrôle*.

Déroulement des appels pour les actions de l'automate compris dans le contrôleur.

```
Client                                          Serveur

NouveauModuleGUI::__init__()
Dash::StateController_i::Configure()
Dash::Controller_i::Configuring()
NouveauModuleController::Configuring()
NouveauModuleGUI::Configuring()  -------->  Dash::StateController::Configure()
                                            Dash::Controller_i::Configuring()
                                            Dash::Controller_i::CheckHardware()
                                            NouveauModule::Controller_i::ConfigureHardware()
```

Voir par exemple le code de ce client graphique : *guiClientCorba.py*

Controlleur graphique

Si le conrôleur (premier serveur CORBA) et le client (premier client CORBA) sont tous les deux écrits dans le même langage, alors on écrira directement leur code au sein d'un même programme afin de se passer d'un appel au bus CORBA.

Par contrôleur on entend à présent :

— un client d'une appliquation.

— une interface graphique

— un serveur CORBA

Voir par exemple le code suivant : *guiControleurCorba.py*

Voici un DIFF soulignant les principales différences avec le code du client graphique donné ci-dessus :

```
37c39,41
< class OnlinecalibratorGUIController(StateController.StateController_i):
---
> class OnlinecalibratorGUIController(Onlinecalibrator__POA.Controller, # Attention!! : l'ordre est
>                                     StateController.StateController_i,
>                                     tcpClient.LedEmbeddedClientSocket):

43a44
>         tcpClient.LedEmbeddedClientSocket.__init__(self)

47,52d55
<     def find_server(self,name):
<         server = Server.Server_i.find_server(self,name)
<         if server is None: return server
<         server = server._narrow(Dash.StateController)
<         return server
<

95a98,114
>     def doDelay(self, valueOn2bytes):
>         self.SendDelay(valueOn2bytes)
>     ...
>

104c123
<         self.server.doDelay(string.atoi(hexaValue, 16))
---
>         self.controller.doDelay(string.atoi(hexaValue, 16))
...

128c147
<     def __init__(self,name = "Onlinecalibrator/GUI"):
---
>     def __init__(self,name = "Onlinecalibrator/Controller"):

132,133c151,152
<         self.set_wmclass('hess_main', 'Onlinecalibrator GUI')
<         self.set_title('Onlinecalibrator GUI')
---
>         self.set_wmclass('hess_main', 'Onlinecalibrator Controller')
>         self.set_title('Onlinecalibrator Controller')

147c166
```

```
<         self.onlinecalibratorframe = gtk.Frame(label = "Onlinecalibrator/GUI")
---
>         self.onlinecalibratorframe = gtk.Frame(label = "Onlinecalibrator Controller")

150,151d168
<         self.onlinecalibratorversion = gtk.Label("Controller not found")
<         self.onlinecalibratorversion.set_name("bold20")

155d171
<         h.attach(self.onlinecalibratorversion,0,2,0,1,gtk.EXPAND|gtk.FILL,gtk.EXPAND|gtk.FILL,5,5)

247,249d262
<
<         self.find_controller()
<

255,257c269,270
<             self.find_controller()
<             if self.server.GetState() == Dash.StateController.Safe:
<                 self.server.Configure()
---
>             if self.controller.GetState() == Dash.StateController.Safe:
>                 self.controller.Configure()

259,260c272
<             self.add_command_bunch(c,self.onlinecalibratorstatus.set_text,`self.server.GetState()`
---
>             self.add_command_bunch(c,self.onlinecalibratorstatus.set_text,`self.controller.GetStat

265c277
<             self.server = None
---
>             self.controller = None

271d282
<             self.find_controller()
...

363,381d364
<
<     def find_controller(self):
<         """Try to find the controller """
<         name = "Onlinecalibrator/Controller"
<
<         self.server = self.controller.find_server(name)
<
<         if self.server is not None:
<             self.server = self.server._narrow(Onlinecalibrator.Controller)
<
<         c = self.create_command_bunch()
<
<         if self.server is None:
<             self.add_command_bunch(c,self.onlinecalibratorstatus.set_text,"N/A")
<             self.add_command_bunch(c,self.onlinecalibratorversion.set_text,"Cannot find object Onl
```

```
<          else :
<              self.add_command_bunch(c,self.onlinecalibratorstatus.set_text,'self.server.GetState()
<              #self.add_command_bunch(c,self.onlinecalibratorversion.set_text,self.server.GetControl
<          self.execute_command_bunch(c)
435a419
>
```

1.3 Socket

1.3.1 Introduction

Utilisation des sockets pour la communication client-serveur.

1.3.2 Client

La socket garde la connexion ouverte durant toute la session. Les sockets n'étant pas prévues à cet effet, il faut envoyer un `\n` pour envoyer effectivement les données.

Telnet

```
$ telnet hostName portNumber
...
Ctrl + ]
> quit
$
```

Python

La socket ouvre une nouvelle connexion à chaque requette. Elle transmet la réponse puis se ferme.

```
#!/usr/bin/python
import sys
import socket

class ClientSocket():

    def __init__(self, ServerHost, ServerPort):

        self.ServerHost = ServerHost
        self.ServerPort = ServerPort
        self.socket = None


    def SendAndRecv(self, message):
        response = "socket error"
        print "%s\t..." % message,

        try:
            self.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            socket.setdefaulttimeout(1)
            self.socket.connect((self.ServerHost, self.ServerPort))
            self.socket.send(message + "\n")

            response = ""
            newResponse = self.socket.recv(8)
            while (newResponse != ""):
                response = response + newResponse
                newResponse = self.socket.recv(8)

            # remove ending \n
            response = response[:-1]

        except socket.error, err:
            response = "%s" % (err)
```

```
        self.socket.close()
        return response
```

1.3.3 Serveur

La solution utilise des fils (au sens fork) plutôts que des fils d'execution (thread) bien que ceux-ci soit plus réactifs.

La solution retenue est d'ouvrir une nouvelle connexion à chaque requette et de fermer la sockette une fois la réponse transmise.

Rien n'a été fait pour gérer la concurrence entre les fils.

1.4 Programmes de contrôle

1.4.1 Introduction

Il s'agit des principaux démons embarqués. Leur rôle est de fournir une API pour les appels aux drivers.
Les principaux démons sont :
— Sam : Banc de test des mémoires analogiques
— Big : Interface principale
— Zora : Gestion des données
— Emilie : Gestion des triggers

1.4.2 Organisation des répertoires

Répertoires communs

Au plus haut niveau :
— `bin` : programme de haut niveau (reliquats de Hess-1)
— `CHANGES` : Changelog
— `Makefile`
— `Make.rules` : Compilation croisée
— `Make.macros` : "
— `Target.rules` : Compilation sur cible
— `Target.macros` : "
Répertoire `Server` :
— `Bigd`
— `Emilie`
— `Client` : simulation des commandes de base depuis la ferme de PC
— `include` : includes propores aux caméras et donc communs à tous les démons
— `Makefile`

Big

— `Big` : interpréteurs de commandes
— `Corba` : envoi des données aquises par la caméra
— `Drawer` : configuration des tirroirs
— `Farm` : représentation de la ferme de PC
— `include`
— `Scheduler` : lance les taches programmées
— `Server` : serveur de sockets
— `ServerUtil` : fonctions des points d'entrées (start/stop...)
— `Util` : misc ?
— `test` : tests
— `BIG-CHANGES` : change log
— `Makefile`

— `Client` : UI reprenant le répertoire *../Client*
— `Socket` : communication des résultats (out) avec un client TCL-TK

1.4.3 Compilation

— Les démons sont compilé en utilisant le compilateur croisé.
— Le chemin vers le compilateur croisé est indiqué dans le fichier *.shl_cross_compile* ; comme indiqué dans le fichier ier *Make.rules* :
```
CROSS_COMPILE   := $(shell cat $(HH_HOME)/SBig/.shl_cross_compile)
```
— La compilation se fait sur le serveur de boot N1N9.
```
[roche@n1n9]~/SBig/Server% make
```

1.4.4 Fonctionnement des serveurs

Dans le diagramme ci-dessous, chaque flèche indique la création d'un thread. Leur priorité est inscrite à côté des flèches. Une étoile complète les flèches lorsque plusieurs threads sont créés.

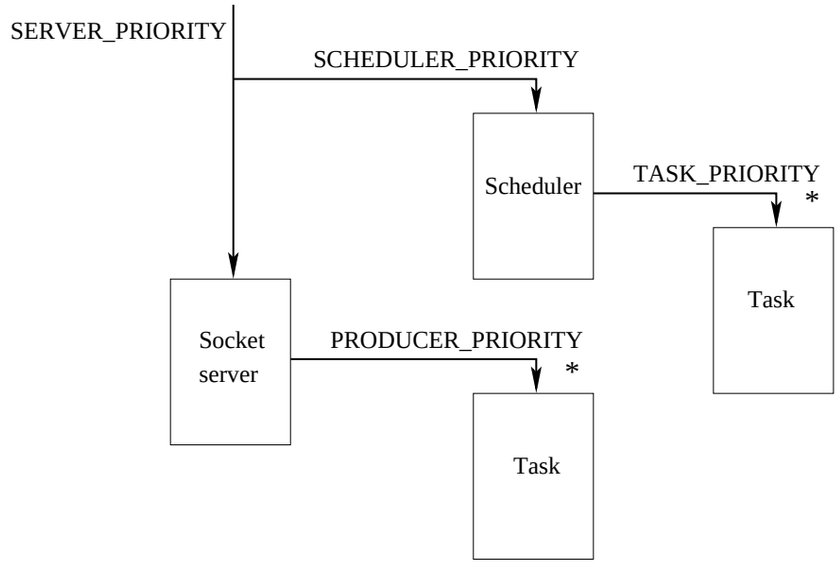


FIGURE 1.3 – Schéma des threads des serveurs

Attention dans hess2 les priorités effectives sont inversées : `thread_prior_max - SERVER_PRIORITY` et donc plus elle sont faibles et plus elle sont prioritaires (comme pour nice).

Précision : le schéma ci dessus est celui des serveurs de monitoring (Big et Bunny). Les serveurs d'acquisition (Zora et Emilie) utilisent quand à eux un scheduleur différent générant uniquement un ou deux autres thread de priorités `SENDER_PRIORITY` (au lieu de `TASK_PRIORITY`).

- les threads privilégiés seront les plus brefs possibles.
- les serveurs de contrôle observe les priorités `SCHEDULER_PRIORITY < SERVER_PRIORITY`. Par conséquent le monitoring est prioritaire sur la réception et l'envoi des ordres.
- les serveurs de données observe les priorités `SERVER_PRIORITY < SCHEDULER_PRIORITY`. Par conséquent la réception des ordres est prioritaire sur l'acquisition.
- La relation `SERVER_PRIORITY < PRODUCER_PRIORITY` permet d'interrompre les requêtes "begin" et garanti le traitement atomique des requêtes "fast_".
- On devrait (je pense) avoir `TASK_PRIORITY < SCHEDULER_PRIORITY` car les tâches de monitoring s'excluent mutuellement et sont ordonnancés les unes à la suite des autres.
- `< SENDER_PRIORITY < SCHEDULER_PRIORITY` favorise l'envoi des donnée par les sockets au vidage des fifos. En effet (j'imagine) que les opérations d'entrées/sorties sur les fifo rendent la main d'elles mêmes en cas d'attente.
- Bien que définis, `SENDER_PRIORITY` et `CONSUMER_PRIORITY` ne sont pas utilisées.

Ectat actuel sur la caméra :

Priorité	Monitoring	Acquisition
Server	9	9
Producer	9	9
Scheduler	10	12
Task	11	
Sender		10, 11

Valeurs proposées pour le module de test :

Priorité	Monitoring	Acquisition
Server	8	8
Producer	9	9
Scheduler	7	12
Task	6	
Sender		10, 11

Voici un programme simplifié permettant de tester cette architecture. A l'édition des liens, il faut indi-

quer la librairie adéquate : `gcc -lpthread simuServer.c`.

Pseudo temps-réel

Sous linux, 3 ordonnancements co-existent.
— OTHER : préemptif avec priorité dynamique (nice)
— RR : temps réel souple (priorité statique) mais préemptif à priorité égale
— FIFO : temps réel souple non préhemptif (comme windows 95)

S'il existe un ou plusieurs processus FIFO ou RR, ils sont séléctionnés en premier. Lors de la phase de débugage, il est important de conserver un shell s'exécutant à un niveau de priorité supérieur. Attention lors du fork, il faut placer la modification de priorité dans le code du processus père.

Thread

Les threads sont des processus allégés ne réclamant que peu de ressources pour les changements de contexte. Chaque thread dispose personnellement d'une pile et d'un contexte d'exécution contenant les registres du processeur et un compteur d'instruction. En revanche, ils se partagent les données statiques et dynamiques.

Sous LINUX, l'implémentation usuelle des threads est effectuée dans l'espace noyau.

Remarque : un thread finissant envoie un signal au thread parent si celui-ci n'est pas trappe le thread ne meurt pas ... seul le systeme le fera disparaitre 1 mn plus tard. Ainsi, pour ne pas borner le nombre de threads lancés par le nombre de threads qu'il est possible de lancer en même temps, il faut soit utiliser la fonction `pthread_join` soit détacher les threads.

Comme pour les processus, on peut modifier l'ordonnancement des threads. Les ordonnancements temps-réel nécessitent un UID effectif nul, sinon la fonction `pthread_create` échouera avec l'erreur EPERM.

Les serveurs embarqués pour HESS utilisent 2 thread, le fils principal devenant alors le second thread :

```
/* privilèges root */
setuid(geteuid());

/* processus privilégié
 * attention, les tread créés sans l'attribut 'PTHREAD_EXPLICIT_SCHED'
 * hériteront de la nouvelle priorité */
paramSocketServer.sched_priority = thread_prior_max - SERVER_PRIORITY;
sched_setscheduler(0, __SCHEDULER_POLICY, &paramSocketServer)

/* construction du thread executant la fonction 'Scheduler' */
pthread_attr_setdetachstate(&threadAttributs, PTHREAD_CREATE_DETACHED)
pthread_attr_setschedpolicy(&threadAttributs, __SCHEDULER_POLICY)
pthread_attr_setinheritsched(&threadAttributs, PTHREAD_EXPLICIT_SCHED)
paramThreadScheduler.sched_priority =  thread_prior_max - SCHEDULER_PRIORITY
pthread_attr_setschedparam(&threadAttributs, &paramThreadScheduler)
pthread_create(&threadScheduler, &threadAttributs, Scheduler, NULL)

/* definition des thread qui éxecuteront la fonction process_message */
pthread_attr_setdetachstate(&thread_attr, PTHREAD_CREATE_DETACHED)
pthread_attr_setschedpolicy(&thread_attr, __SCHEDULER_POLICY)
pthread_attr_setinheritsched(&thread_attr, PTHREAD_EXPLICIT_SCHED)
scheduler._parameters.sched_priority = thread_prior_max - BIG_SCHEDULER_PRIORITY
pthread_attr_setschedparam(&thread_attr, &scheduler._parameters)

/* rendre les privilèges */
setuid(realUserIdentifier);

/* lancement du serveur de socket */
tcp_server();
```

Socket

Les serveurs embarqués pour HESS utilisent un schéma séquentiel.

```
/* creation de la socket standardiste */
sock_contact = socket(AF_INET, SOCK_STREAM, 0)
bind(sock_contact, ...)
listen(sock_contact, ...)
```

```
/* connexion sequentielle aux clients */
while(loop) {
    sock_dialog = accept(sock_contact, ...)
    process_connection(sock_dialog)
```

Ce schéma sequentiel peut alors déboucher sur une prise en charge parallèle :

```
if (...)
{
  /* traitement sequentiel */
  ...
}
else
{
/* thread executant la fonction 'process_message' avec en paramètre la socket */
  pthread_create(&thread_producer[n], &producer__attr, process_message, (void *)sock)))
}
```

Remarque : `CLOSE_WAIT` is when the other endpoint have closed the connection and the kernel is vaiting for the application to confirm the close by closing the socket descriptor. There is no timeout for `CLOSE_WAIT`. It persists for as long as the application has not closed the socket.

```
# netstat -nap
Proto Recv-Q Send-Q Local Address               Foreign Address             State       PID/Program
tcp       55      0 192.168.1.169:1806          134.158.155.234:53291       CLOSE_WAIT  -
```

1.4.5 Grammaire des controleurs

— Big

```
orders: SHUT_
      | PING_
      | preposition stanza ENDMSG
      | stanza ENDMSG

preposition: BEGIN
           | FAST_

stanza: order
      : stanza order

order: CAMERA camera
     | NODE farm
     | FIFO fifo
     | DRAWER drawer
     | DAEMON daemon
     | RUN run
     | TASK task
     | INI drawer2
     | TRG
     | ACK FIFODATA_EMPTY
     | ACK FIFODATA_PROBE_EVTSIZE
     | ???
     | ...
```

— Camera

```
camera: CONFIGURE
      | SAFE
      | RUN
      | STOP
      | FIFODATA_PROBE_EVTSIZE
```

— Farm

```
farm: CONTROLLER node
    | MONITORING node
    | ADD 0x%4X%n node
    | CURRENT %d
    | DELETE %d
    | DISABLE %d
    | ENABLE %d
    | FREQUENCY %d
    | LIST

node: %s %s %s%n { %x%n}+
```

— Fifo

```
fifo: RESET
    | RESET_INTERFACE
    | INIT_BOXBUS
    | RESET_WFIFO
    | RESET_RFIFO
```

— Drawer

```
drawer: TARGET
      | SET_CNTRL_SLC
```

```
        | SET_CNTRL_DAQ
        | SET_CNTRL_L2
        | SET_CNTRL_SAM_DAQ
        | SET_CNTRL_SAM_ND
        | SET_CNTRL_SAM_REGISTER
        | SET_CNTRL_TRIGGER
        | SET_TRIGGER_DISABLE
        | SET_CNTRL_VOLTAGE
        | SET_HV_OFF
        | SET_VOLTAGE_VALUE
        | SET_VOLTAGE_LEVEL
```

— Daemon

```
daemon: DEBUG %X
      | CLEARSCREEN
      | INIT_DAQ
```

— Run

```
run: START
   | STOP
   | INIT
```

— Task

```
task: STATUS
    | LIST
    | UNLOCK
    | ADD
    | SCHEDULER_DELAY %d
    | FREQUENCY %d
    | DELETE
    | ENABLE
    | DISABLE
```

— Drawer2

```
drawer2: cmdId label ...
       |
       |
       |
       | ...
```

1.5 Drivers

1.5.1 Introduction

Les drivers sont directement compilés sur la machine cible.

Chapitre 2

Outils

Matériel et adresses IP :

— **DNS**
  — `134.158.152.55` DNS par relais (cf named et dnsmasq => system-config-bind)
  — `134.158.152.146` DNS lpnhe
  — `134.158.69.191` DNS cc à Lyon
— **Réseau privé**

  Ajouter une route vers le réseau HESS :

```
iface eth0 inet dhcp
        up route add -net 192.168.1.0 netmask 255.255.255.0 gw 134.158.152.55
        down route del -net 192.168.1.0 netmask 255.255.255.0 gw 134.158.152.55
```

  Organisation des plages d'adresses :
  — n1n3 (alias lpnp90) : serveur DNS, serveur de NFS des */home* et passerelle.
  — n1n9 : serveur de boot
  — n1n40..n1n41 : pizza 64 sous OS FEDORA
  — n1n50..n1n59 : switchs
  — n1n60..n1n69 : serveurs TX
  — n1n80..n1n89 : alimentations
  — n1n160..n1n169 : processeurs POWERPC embarqués sur les caméras

  Ceci correspond à ce que l'on pourra trouver dans le fichier *lpnp90.in2p3.fr :/etc/hosts* à default d'accéder au DNS

```
192.168.1.1     n1n1.in2p3.fr   n1n1   cleo
192.168.1.2     n1n2.in2p3.fr   n1n2
192.168.1.3     n1n3.in2p3.fr   n1n3   lpnp90  134.158.152.55 # fc7 (PIII x2)
192.168.1.4     n1n4.in2p3.fr   n1n4   lpnp199 # fc8 (Xeon x4)
192.168.1.5     n1n5.in2p3.fr   n1n5   lpnp167 # fc9 (PIV x2)
192.168.1.6     n1n6.in2p3.fr   n1n6   # fc9 (PIV x2)
192.168.1.7     n1n7.in2p3.fr   n1n7
192.168.1.8     n1n8.in2p3.fr   n1n8   moto
192.168.1.9     n1n9.in2p3.fr   n1n9   lpnpsv165 # fc12 (Xeon x8)
192.168.1.10    n1n10.in2p3.fr  n1n10  velo
192.168.1.11    hess01.in2p3.fr n1n11  hess01
192.168.1.12    jade.in2p3.fr   n1n12  jade
192.168.1.13    n1n13.in2p3.fr  n1n13
192.168.1.14    n1n14.in2p3.fr  n1n14  # n6n1 fc7 (PIII boot camera 20 tiroirs)
192.168.1.15    n1n15.in2p3.fr  n1n15  # fc12 passwdInfo
192.168.1.16    n1n16.in2p3.fr  n1n16  # fc6 (PIV)
192.168.1.17    n1n17.in2p3.fr  n1n17
192.168.1.18    n1n18.in2p3.fr  n1n18
192.168.1.20    n1n20.in2p3.fr  n1n20  # fc11 (Amd64 x2) 134.158.155.251
192.168.1.21    n1n21.in2p3.fr  n1n21
192.168.1.22    n1n22.in2p3.fr  n1n22
192.168.1.30    n1n30.in2p3.fr  n1n30  # fc9 (salle camera, Julien Bolmon)
192.168.1.31    n1n31.in2p3.fr  n1n31  # fc12 (salle camera, Julien Bolmon)
192.168.1.32    n1n32.in2p3.fr  n1n32
192.168.1.33    n1n33.in2p3.fr  n1n33  # linux2.4 (PIII x2)
192.168.1.40    n1n40.in2p3.fr  n1n40  # fc9 pizza 64
192.168.1.41    n1n41.in2p3.fr  n1n41  # fc9 pizza 64 (salle caméra)
192.168.1.42    n1n42.in2p3.fr  n1n43  # fc9 (PC double écran)
192.168.1.43    n1n42.in2p3.fr  n1n43  # fc9 (PC double écran Namibie)

192.168.1.50    n1n50.in2p3.fr  n1n50  # osciloscope LeCroy
192.168.1.51    n1n51.in2p3.fr  n1n51  # switch 8 ports
192.168.1.52    n1n52.in2p3.fr  n1n52  # switch 8 ports
192.168.1.53    n1n53.in2p3.fr  n1n53  # switch 16 ports
```

```
192.168.1.100   n1n100.in2p3.fr n1n100
192.168.1.101   n1n101.in2p3.fr n1n101
192.168.1.102   n1n102.in2p3.fr n1n102
192.168.1.103   n1n103.in2p3.fr n1n103
192.168.1.104   n1n104.in2p3.fr n1n104
192.168.1.105   n1n105.in2p3.fr n1n105
192.168.1.106   n1n106.in2p3.fr n1n106
192.168.1.107   n1n107.in2p3.fr n1n107
192.168.1.108   n1n108.in2p3.fr n1n108
192.168.1.109   n1n109.in2p3.fr n1n109
192.168.1.110   n1n110.in2p3.fr n1n110

192.168.1.112   n1n112.in2p3.fr n1n112
192.168.1.137   n1n137.in2p3.fr n1n137

192.168.1.158   n1n158.in2p3.fr n1n158  camera8  # ??
192.168.1.159   n1n159.in2p3.fr n1n159  camera9  # ??
192.168.1.160   n1n160.in2p3.fr n1n160  camera10 # ??
192.168.1.161   n1n161.in2p3.fr n1n161  camera11 # gigabit zora sur bigcamera
192.168.1.162   n1n162.in2p3.fr n1n162  camera12 # carte zora sur bigcamera
192.168.1.163   n1n163.in2p3.fr n1n163  camera13 # carte émilie sur bigcaméra
192.168.1.164   n1n164.in2p3.fr n1n164  camera14 # ???
192.168.1.165   n1n165.in2p3.fr n1n165  camera15 # ???
192.168.1.166   n1n166.in2p3.fr n1n166  camera16 # carte big sur bigcamera
192.168.1.167   n1n167.in2p3.fr n1n167  camera17 # carte bunny sur bigcamera
192.168.1.168   n1n168.in2p3.fr n1n168  camera18 # carte banc de test
192.168.1.169   n1n169.in2p3.fr n1n169  camera19 # carte module 20 tirroirs

192.168.1.170   n1n170.in2p3.fr n1n170  arm01 # Systeme de calibration

192.168.1.246   n1n246.in2p3.fr n1n246  momo
192.168.1.247   n1n247.in2p3.fr n1n247  sharon  sharon.in2p3.fr
192.168.1.248   n1n248.in2p3.fr n1n248  camera0
192.168.1.249   n1n249.in2p3.fr n1n249  pluton  pluton.in2p3.fr
192.168.1.250   n1n250.in2p3.fr n1n250  camera2
192.168.1.251   n1n251.in2p3.fr n1n251
192.168.1.252   n1n252.in2p3.fr n1n252  camera3
192.168.1.253   n1n253.in2p3.fr n1n253  hades   hades.in2p3.fr
192.168.1.254   n1n254.in2p3.fr n1n254  camera4

192.168.4.1     n4n1.in2p3.fr   n4n1
192.168.4.2     n4n2.in2p3.fr   n4n2
192.168.4.3     n4n3.in2p3.fr   n4n3
192.168.4.4     n4n4.in2p3.fr   n4n4
192.168.4.5     n4n5.n2p3.fr    n4n5
192.168.4.6     n4n6.n2p3.fr    n4n6
192.168.4.7     n4n7.in2p3.fr   n4n7
192.168.5.1     n5n1.in2p3.fr   n5n1
192.168.5.2     n5n2.in2p3.fr   n5n2
192.168.6.1     n6n1.in2p3.fr   n6n1
192.168.6.2     n6n2.in2p3.fr   n6n2    momo
192.168.6.3     n6n3.in2p3.fr   n6n3
192.168.6.3     n6n4.in2p3.fr   n6n4
```

```
192.168.6.4     n6n5.in2p3.fr   n6n5
192.168.6.6     n6n6.in2p3.fr   n6n6
192.168.6.11    n6n11.in2p3.fr  n6n11
192.168.6.12    n6n12.in2p3.fr  n6n12
192.168.6.13    n6n13.in2p3.fr  n6n13
192.168.6.14    n6n14.in2p3.fr  n6n14
192.168.6.80    n6n80.in2p3.fr  n6n80 wiener

134.158.155.140 lpnp344.in2p3.fr lpnp344
134.158.155.201 lpnp407.in2p3.fr lpnp407
134.158.152.146 lpnsu9.in2p3.fr nsrhost
196.44.140.213  paris.hess.na   paris

196.44.140.214  base.hess.na

134.158.152.218 lpnhess1
134.158.152.219 lpnhess2
```

— **Banc de test**
  — chassis 21 slots en test
— **Module 20 tiroirs**
  — Boitier LED.
  — Le banc de test **ne doit plus** être utilisé pour les développement informatiques.
  — Le compte officiel a utiliser est le compte *camera*. Pour lancer le controleur caméra :
  ```
  $ su - hess
  $ cd $HESSUSER/camera2
  $ ./start-camera5
  ```
— **Caméra**
  Intégration des Softs petit à petit :
  — Le compte officiel a utiliser est le compte *bigcamera*.
  — 4 cartes controleurs.
  — 2 Liens gigabit via la Pizza Box 192.168.7.41
  — Timestamp + gps à intégrer
  — lancer la caméra :
  ```
  $ login 'camera' ou 'hess' sur lpnp90
  $ cd ~hess
  $ tcsh
  $ source env-csh
  $ cd .../camera2
  $ ./start-camera88
  ```
— **Controlleurs** CORBA
  — Le compte officiel a utiliser est le compte *bigcamera*.
— **Sauvegardes sur** *n1n9 :/data/home90.backup/*
  Cf les fichiers :
  — */etc/cron.d/backup-cvs.cron*
  — */etc/cron.d/backup-home.cron*
  Arrêté par Mathieu suite à la saturation du disque.
— **lien symbolique vers les Partitions racines des caméra** Le lien */opt/sugarhat/devkit* est utlisé par les serveurs TFTP (*/tftpboot/**) et NFS (*/etc/exports*).
— **NIS** :
  `cd /var/yp ; make`
  cf */etc/nsswitch.conf* : ordre de priorité (txt, db... dns)
  configuration des postes clients (suppression des fichiers locaux...)
  Changer de mot de passe :
  `$ yypasswd`

Créer un compte :

```
n1n3# /usr/sbin/adduser camera2
n1n3# /usr/sbin/adduser camera2_64bits
n1n3# chmod og+rx /home/camera2
n1n3# chmod og+rx /home/camera2_64bits
n1n3# vi /etc/passwd
camera2:x:10286:220::/home/camera2:/bin/zsh

n1n3# chown -R camera2.camera2 /home/camera2_64bits
n1n3# cd /var/yp
n1n3# make

// sur les postes GUI :
n1n40# ln -s /mnt/homes/lpnp90/camera2 /home/camera2
n1n43# ln -s /mnt/homes/lpnp90/camera2 /home/camera2
n1n43# su - camera2

// sur les postes Serveur-embarqué :
n1n9#  ln -s /mnt/homes/lpnp90/camera2_64bits /home/camera2
camera10# vi /etc/fstab.local
192.168.1.3:/home/camera2_64bits   /home/camera2    nfs defaults        0 0
```

Afficher les fichiers gérés par NIS :

```
$ ypcat passwd
$ ypcat hosts
$ ypcat services
```

— **Modifier le réseau et les routes sous REDHAT**

```
# system-config-network
```

— **Passer en mode routeur sous REDHAT**

```
# vi /etc/sysctl.conf
# /sbin/sysctl -p      # (reload)
```

— **backup sur n1n3**

```
Il faut lancer "nwrecover" sur lpnp90,
sélectionner une date antérieure (Options=> Change Browse Time),
sélectionner le fichier, et lancer la récupération. Ca prend du temps...
```

2.1 CVS

2.1.1 Introduction

Les codes sources sont archivés sur 2 serveurs CVS :
— un sur *lpnp90.in2p3.fr*

```
[roche@lpnp90]~% ls /home/cvsroot
AGN                kaskade            parisanalysisuser  SBig
analysis           kaskade_cat        parispointing      Sdaq
corsika            kaskade_cpp        parisprograms      Seminaires
corsika-6600       ledcontrol         parisreco          SeminaireWeb
CVSROOT            mathutils          parisspectrum      simudb
DBase              MemAna             parisusers         spectrum
doc                model3D            PersoJpt           thesejulien
globularclusters   modelisation       PersoLoic          thesemarianne
gpib_instruments   ModelWeb           PersoMathieu       thesenicolas
gtkextra           morphology         PointingWeb        trigger
hessana            neural             Pydaq              triggerl2
HESSFpga           onlinecalibrator   rfio               usb_instruments
HESSIINote         owis               RootTest           waverunner
HESSPat            papers             samguide           webcam
hessphp            parisanalysis      samtest
```

— l'autre hébergé à Berlin (pour le code de camera2 par exemple)
Les principaux projets sont
— *Sdaq* : serveur HESS 1
— *SBig* : serveurs HESS2

2.1.2 Commandes

Checkout

— Pour télécharger les sources d'un module

```
$ cvs -d :pserver:roche@lpnhess.in2p3.fr:2401/data/services/cvsroot login
$ cvs -d :pserver:roche@lpnhess.in2p3.fr:2401/data/services/cvsroot co SBig
```

— Pour télécharger uniquement un répertoire d'un module

```
$ mv SBig/Driver/TL2new /TL2newBof
$ cvs -d :pserver:guevara@lpnp90.in2p3.fr:/home/cvsroot co SBig/Driver/TL2new
```

— Permet aussi de récupérer une version tagguée :

```
$ cvs -d :pserver:roche@lpnp90.in2p3.fr:/home/cvsroot co -r drawertest-20090907 SBig
```

— ou encore la version à telle date :

```
$ cvs -d :pserver:roche@lpnp90.in2p3.fr:/home/cvsroot co -D "2009-10-26 10:00" SBig
```

Update

Permet de récupérer la dernière version.
Remarquez le `-n` qui permet de simuler la requête et donc de voir quels fichiers seront ajournés.

```
$ cvs -n update -d
M Server/Bigd/Big/Makefile
U Server/Bunny/Server/ServerStop.c
...

$ cvs status -v Server/Bigd/Big/Makefile
   Status: Locally Modified
   Working revision:    1.5
   Repository revision: 1.5
```

```
$ cvs status -v Server/Bunny/Server/ServerStop.c
   Status: Needs Patch
   Working revision:    1.5
   Repository revision: 1.6

$  cvs diff -r 1.6 Server/Bunny/Server/ServerStop.c
...

$ cvs update -d #(se mettre à jour)

$  cvs -d :pserver:roche@lpnp90.in2p3.fr:/home/cvsroot commit #(commit avec un autre compte cvs)
$  cvs update #(maj formelle non requise avec le même compte : cvs -n update n'affiche plus aucune d
```

Remarque : `-d` [à la fin] permet d'intégrer les nouveau répertoires commités.

Diff

Permet de voir les fichiers ont été modifié depuis le dernier commit.

```
$ cvs diff
```

Commit

Remarquez que la requête ci-dessous peut être faite depuis le répertoire d'un autre utilisateur, à condition toutefois d'ignorer les erreurs d'écriture dans ses répertoires `CVS` trouvés à partir du répertoire courant. Ceci permet par exemple de commiter le code de Jean-luc Panazol.

```
$ cvs -d :pserver:roche@lpnp90.in2p3.fr:/home/cvsroot update
$ cvs -d :pserver:roche@lpnp90.in2p3.fr:/home/cvsroot commit
```

Tag

```
$ cd ~/SBig
$ cvs status -v Makefile
   Existing Tags:
        drawertest-20090525             (revision: 1.4)
        DrawerTest_200902               (revision: 1.4)
        drawer_2009_15                  (revision: 1.4)

$ cvs tag drawertest-20090907 .
$ cvs status -v Makefile
   Existing Tags:
        drawertest-20090907             (revision: 1.4)
        drawertest-20090525             (revision: 1.4)
        DrawerTest_200902               (revision: 1.4)
        drawer_2009_15                  (revision: 1.4)

$ cd /tmp
$ cvs co -r drawertest-20090907 SBig
```

2.1.3 Editing administrative files

```
You edit the administrative files in the same way that you would edit any other module.
Use 'cvs checkout CVSROOT' to get a working copy, edit it, and commit your changes in the normal way
```

2.1.4 FAQ

— `No space left on device`
```
$ cvs commit
...
unable to write, file ...
No space left on device
cvs commit: cannot close temporary file /tmp/cvsSCwP1t: No space left on device
```
CVS passe par l'espace temporaire */tmp* de la partition <*racine*> avant d'ajouter les fichiers dans la partition */services*.

— `**** Access denied: Insufficient Karma`
Je ne peux plus faire de commit sur mes nouveaux projets, j'ai testé de modifier en vain mes droits d'accès.
```
$ cvs commit
cvs commit: Examining .
**** Access denied: Insufficient Karma (roche|bof)
cvs commit: Pre-commit check failed
...
```
Mathieu : Les droits d'écriture s'octroient module par module via les fichiers d'administration du CVS : fichiers *avail* et *modules* du module CVSROOT.

— CVS refuse d'ajouter un fichier binaire.
C'est normal car il ne peut pas faire grand chose pour les merge et surtout il rique d'y modifier -entre autre- les chaînes `$Id$`. Il faut lui préciser que c'est bien un binaire. (cf info)
```
  The `-kb' option available with some CVS commands insures that
neither line ending conversion nor keyword expansion will be done.

$ cvs add -kb -m "libnetsnmp.so (binary)" libnetsnmp.so
$ cvs commit libnetsnmp.so
```

2.2 TFTP

2.2.1 Introduction

Les cartes embarquées boot via TFSP en téléchargeant les noyaux hébergés par le serveur **n1n9**.

2.2.2 n1n9 : serveur de boot

Les noyaux sont stockés dans le répertoire */tftboot*. Il y a un noyau par caméra : chaque noyau est suffixé par le numéro de la caméra.

Le serveur TFSP issu du paquetage *tftp-server-0.48-6.fc9.x86_64* est configuré par le fichier */etc/xinetd.d/tftp* :

```
service tftp
{
        socket_type             = dgram
        protocol                = udp
        wait                    = yes
        user                    = root
        server                  = /usr/sbin/in.tftpd
        server_args             = /tftpboot
        disable                 = no
        per_source              = 11
        cps                     = 100 2
        flags                   = IPv4
}
```

Les noyaux linux sont rangés à l'aide de liens symboliques :

```
$ ls -l /tftpboot
...
zImage12 -> /opt/CES/devkit/target12/usr/src/linux/arch/powerpc/boot/zImage.rio
...
```

2.2.3 Carte cliente

La carte cliente est configuré à l'aide de les commandes `set inet` et `set boot` afin de pointer sur le serveur TFSP.

```
procmon> show inet
procmon> show boot

procmon> set boot_devive
le                              # local ethernet

procmon> set boot_filename
/tftpboot/zImage                # filename in tftp for kernel image

procmon> show inet_boot_server
192.168.1.9                     # TFTP server
```

2.3 NFS

2.3.1 Introduction

Chacun des noyaux (associé à une caméra) va monter son propre système de fichier racine via NFS. Les home de chacun des utilisateurs seront également montés via NFS sur chaque partition racine.

Il y a 2 cathégories de montages répartis sur 2 servers NFS :

— **n1n3** répertoires homes des utilisateurs
— **n1n9** partitions racine des cartes embarquées

2.3.2 n1n3 : serveur *home*

Partitions exportées : cf */etc/exports*

— `/data`
— `/diske`
— `/diskf`
— `/home`

2.3.3 n1n9 : serveur des partitions racines

— Partitions exportées : cf */etc/exports*

```
/opt/sugarhat/devkit/target *(rw,no_root_squash,no_all_squash)
/opt/sugarhat/devkit/target18 *(rw,no_root_squash,no_all_squash)
/opt/sugarhat/devkit/target19 *(rw,no_root_squash,no_all_squash)
/opt/sugarhat/devkit/target17 *(rw,no_root_squash,no_all_squash)
#/opt/sugarhat/devkit/target8 *(rw,no_root_squash,no_all_squash)
/opt/CES/devkit/target8 *(rw,no_root_squash,no_all_squash)
/opt/sugarhat/devkit/target9 *(rw,no_root_squash,no_all_squash)
```

— Les partitions racines sont rangés à l'aide de liens symboliques :

```
$ ls -l /opt/CES/devkit
...
target12 -> eldk/ppc_74xx_20080904_patched_c12
...
```

Attention : changer lien symboliques ne suffit pas pour modifier la partition exportée.

```
# tail /var/log/messages
May 27 16:13:25 n1n9 mountd[28107]: refused mount request from 192.168.1.162 \
for /data/CES/shl-3.2.1/eldk-4.0/ppc_74xx_20080904_c12_rpm (/): not exported
```

— Prise en compte des modifications :

```
# /etc/init.d/nfs reload
```

— Vérification :

```
# /usr/sbin/exportfs
/data/CES/shl-3.2.1/eldk-4.0/ppc_74xx_20080904_patched_c18
                <world>
/data/CES/shl-3.2.1/eldk-4.0/ppc_74xx_20080904_patched_c17
                <world>
/data/CES/shl-3.2.1/eldk-4.0/ppc_74xx_20080904_patched_c13
                <world>
/data/CES/shl-3.2.1/eldk-4.0/ppc_74xx_20090217_c19
                <world>
/data/CES/shl-3.2.1/eldk-4.0/ppc_74xx_20090217
                <world>
/data/CES/shl-3.2.1/eldk-4.0/ppc_74xx_nroche
```

— **Attention** à conserver les droits d'accès aux fichier si vous duppliquez une partition racine.

```
# cp -Rp ppc_74xx_20080904_patched_c12 ppc_74xx_c12_test
```

— Cf le script de Richard : *n1n9 :/data/CESmake_clone_target.sh*.

```
#!/bin/bash
```

```
cd /opt/CES/devkit/eldk || exit 1
LIST='17 18 19'
[ $# -ne 0 ] && {
        LIST="$*"
}
for camera in $LIST; do
        source=$(file ppc_74xx | awk -F'to' '{print $2}' | sed -e "s/'//" -e 's/ `//')
        [ -d $source ] || break
        if [ ! -d ${source}_c$camera ]; then
                echo "Copying $source .. ${source}_c$camera"
                cp -af $source ${source}_c$camera
                echo "Linking ${source}_c$camera --> ../target$camera"
                rm -f ../target$camera 2>/dev/null
                ln -s /opt/CES/devkit/eldk/${source}_c$camera /opt/CES/devkit/target$camera
        else
                echo "Rsync-ing $source/ .. ${source}_c$camera/"
                rsync -auv $source/ ${source}_c$camera/
        fi
        grep "target$camera" /etc/exports || {
                echo "/opt/sugarhat/devkit/target$camera *(rw,no_root_squash,no_all_squash)" \
                >> /etc/exports
        }
done
/etc/init.d/nfs reload
showmount -e localhost
```

2.3.4 Import des homes sur n1n9

2.3.5 Import des homes sur les cartes processeur

Ajout du */home* personnel sur la caméra **camera19**

— fichier */opt/CES/target16/etc/fstab.local* :

```
none                              /dev/pts        devpts  defaults        0 0
192.168.1.3:/home/huppert         /home/huppert   nfs     defaults        0 0
192.168.1.3:/home/panazol         /home/panazol   nfs     defaults        0 0
192.168.1.3:/home/hess            /home/hess      nfs     defaults        0 0
192.168.1.3:/home/vincentp        /home/vincentp  nfs     defaults        0 0
192.168.1.3:/home/camera          /home/camera    nfs     defaults        0 0
192.168.1.3:/home/guevara         /home/guevara   nfs     defaults        0 0
192.168.1.3:/home/roche           /home/roche     nfs     defaults        0 0
192.168.1.3:/home/tavernet        /home/tavernet  nfs     defaults        0 0
192.168.1.3:/home/bigcamera_64bits /home/bigcamera  nfs defaults        0 0
```

— fichier */opt/CES/target16/etc/passwd* :

```
...
guevara::10147:220::/home/guevara:/bin/bash
roche::10148:220::/home/roche:/bin/bash
tavernet::225:220::/home/tavernet:/bin/bash
bigcamera::10149:220::/home/bigcamera:/bin/bash
```

— créer le répertoire utilisateur :

```
# mkdir /opt/CES/target16/home/roche
# chown roche.hess /opt/CES/target16/home/roche
```

— redémarer le démon `netfs` sur la carte processeur afin de générer le nouveau fichier *fstab (en fait ça ne marche pas, il faut rebooter)*

```
camera16:~# /etc/init.d/netfs reload
```

— c'est tout
```
$ telnet -l roche camera16
```

2.3.6 Import des partitions racines

La carte cliente est configuré à l'aide de les commandes `set inet` et `set boot` afin de pointer sur le serveur TFSP.

```
procmon> set boot_device
le                                  # local ethernet

procmon> set boot_rootfs
nfs                                 # mount NFS root filesystem

procmon> set boot_rootfs
nfs

procmon> show inet_mount
192.168.1.9:/opt/CES/devkit/target8
```

A priori, il s'agit là d'un moyen de spécifier les paramètres NFS au noyau. Celà revient je pense à modifier la ligne de commande `boot_line` en s'inspirant de la documentation du noyau linux (*Documentation/nfsroot.txt*.

2.4 DHCP

2.4.1 Introduction

Certaines cartes configure leur réseau via le serveur DHCP hébergé par le serveur **n1n9**.

2.4.2 Configuration du serveur sur n1n9

Le serveur DHCP issu -à priori- du paquetage *dhcp-4.0.0-22.fc9.x86_64* est configuré par le fichier */etc/dhcpd.conf* :

```
#
# DHCP Server Configuration file.
#   see /usr/share/doc/dhcp*/dhcpd.conf.sample
#   see 'man 5 dhcpd.conf'
#

#### DHCP HESS Lpnhe Jussieu

ddns-domainname "dhcp";
ddns-update-style none;
ddns-updates off;
authoritative;

##Temps de lease par defaut 1 journée
default-lease-time 86400;
max-lease-time 86400;

option routers 192.168.1.3;
option domain-name-servers 134.158.152.146, 134.158.69.191;
option domain-name "in2p3.fr";
option broadcast-address 192.168.1.255;
use-host-decl-names on;
#use-host-decl-names off;

subnet 192.168.1.0 netmask 255.255.255.0 {

## Clients
pool {
range 192.168.1.160 192.168.1.170;
host arm01      { hardware ethernet 4E:A0:7F:C0:7D:72; fixed-address 192.168.1.170  ;} # arm01
}}
```

2.4.3 Configuration des client 'DEBIAN'

La configuration des clients apparentés au système d'opération DEBIAN se trouve dans le fichier */etc/network/interfaces* :

```
auto eth0
iface eth0 inet dhcp
```

2.5 makefile

2.5.1 Introduction

Cette section décrit les Makefiles utilisés pour compiler :
— le module CVS *SBig*
— les contrôleurs CORBA

2.5.2 Contrôleurs CORBA

Makefile utilisé pour compiler les contrôleurs CORBA.

2.5.3 Module CVS *SBig*

Makefile utilisé pour compiler le module CVS *SBig*.
— Comme le montre la variable `CROSS_COMPILE`, on utilise le même compilateur que celui que l'on a utilisé pour compiler le noyau de la cible.

```
$ grep ^CC /opt/sugarhat/devkit/eldk/ppc_74xx/usr/src/linux/Makefile
CC              = $(CROSS_COMPILE)gcc

$ find ~/SBig/ -name 'Make.*' -exec grep CROSS_COMPILE {} /dev/null \;
./Make.rules:   CROSS_COMPILE   := $(shell cat $(HH_HOME)/SBig/.shl_cross_compile)
./Make.rules:   CROSS_COMPILE   := $(shell cat $(HH_HOME)/SBig/.hhl_cross_compile)
./Make.rules:CC              = $(CROSS_COMPILE)gcc
./Make.rules:STRIP              = $(CROSS_COMPILE)strip

$ cat ~/SBig/.shl_cross_compile
/opt/sugarhat/devkit/eldk-4.0/usr/bin/ppc_74xx-

$ ls -di /opt/sugarhat/devkit/eldk-4.0/usr/bin
53412066 /opt/sugarhat/devkit/eldk-4.0/usr/bin

$ grep CROSS_COMPILE /opt/sugarhat/devkit/sugarhat-env-current.sh
export CROSS_COMPILE="/opt/CES/devkit/eldk/usr/bin/powerpc-linux-"

$ ls -di /opt/CES/devkit/eldk/usr/bin/
53412066 /opt/CES/devkit/eldk/usr/bin/
```

— A priori le Makefile est chargé de reliquats du passé. Par exemple : la variable `MYHAT` ne semble plus être utilisée qu'avec la valeur `sugarhat`

```
$ find . -name 'Make.*' -exec grep MYHAT -A4 {} /dev/null \;
./Make.rules:MYHAT              = sugarhat
...
--
./Make.rules:ifeq ($(MYHAT),sugarhat)
./Make.rules-INCLUDES   += -I/opt/sugarhat/linux-kernel/include
./Make.rules-else
./Make.rules-INCLUDES   += -I/opt/sugarhat/devkit/lsp/ces-rio/linux-2.2.12-2-rio/include
./Make.rules-endif
--
./Make.rules:ifeq ($(MYHAT),sugarhat)
./Make.rules-   CROSS_COMPILE   := $(shell cat $(HH_HOME)/SBig/.shl_cross_compile)
./Make.rules-else
./Make.rules-   CROSS_COMPILE   := $(shell cat $(HH_HOME)/SBig/.hhl_cross_compile)
./Make.rules-endif

$ ls /opt/sugarhat/linux-kernel/include | wc -l
```

```
45

$ ls /opt/sugarhat/devkit/eldk-4.0/usr/bin/ppc_74xx-* | wc -l
21

$ ls /opt/sugarhat/devkit/lsp/ces-rio/linux-2.2.12-2-rio/include
Aucun fichier ou dossier de ce type

$ ls /opt/hardhat/devkit/ppc/7xx/bin/ppc_7xx-*
Aucun fichier ou dossier de ce type
```

— Mode non-verbeux sur les sous-répetrtoires

```
$ $ find . -name "Make*" -exec grep -n '$(MAKE) -s' {} /dev/null \;
./Makefile:34:          $(MAKE) -s -C $$dir && res=1; \
./Server/Bigd/Makefile:60:              $(MAKE) -s -C $$dir && res=1; \
./Server/Makefile:21:           $(MAKE) -s -C $$dir && res=1; \
./Server/Emilie/Makefile:32:            $(MAKE) -s -C $$dir && res=1; \

$ man make
...
 -s   Opération silencieuse ; ne pas afficher les commandes quand elles
            sont exécutées.
```

Compilation des drivers

On compile directement sur la cible.

```
# Take care, this make file will be re-call by kernel's Makefile with M defined

ifndef M
# only first call of this Makefile enter here

.PHONY: all clean

all:
        make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
        make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean

else

# only kernel call of this Makefile enter here.
# warning, $(PWD) env variable is usabble but not $(shell env) which value
# is the directory where make -C was launch
include $(PWD)/../../ppc.rules

EXTRA_CFLAGS += $(CFLAGS) -Wall -Wundef -Wstrict-prototypes -Wno-trigraphs \
          -fno-strict-aliasing -fno-common \
          -ffreestanding \
          -I. \
          -I$(PWD)/../include \
          -I/usr/src/linux/include \
          -I/usr/include/ces
```

```
KBUILD_NOPEDANTIC := 1
DRIVER  := papat26_slc papat26_dat
obj-m   += papat26_slc.o papat26_dat.o

endif
```

Cross-compilation des démons

— L'astuce est là :
```
CROSS_COMPILE   := $(shell cat $(HH_HOME)/SBig/.shl_cross_compile)
```
— MAKEDEPEND complète le make file
```
# apt-get install xutils-dev
$ makedepend toto.c
$ cat Makefile

toto:   toto.c tata.h
        gcc toto.c -o toto

# DO NOT DELETE
toto.o: tata.h titi.h
```

Cross-compilation des démons avec bibliothèque

Ajout de la bibliothèque *libnetsnmp.so*.

— Ajout des fichiers include :
```
$ ssh roche@192.168.1.9
$ cd ~/SBig/Server/Bigd/Util
$ make Wiener.o
/home/roche/SBig/Server/include/Wiener.h:16:38: error: net-snmp/net-snmp-config.h:
No such file or directory

$ cp -fr /opt/sugarhat/devkit/target12/usr/local/include/net-snmp ~/SBig/Server/include/
$ make Wiener.o
```
— Modification du Makefile (*/SBig/Server/Bigd/Makefile*) :
```
LIBS   += -L../lib -lm -lpthread -lnetsnmp
```
— Ajout du fichiers LIBNETSNMP.SO :
```
$ cd ~/SBig/Server/Bigd
$ make big
cannot find -lnetsnmp

$ cp /opt/sugarhat/devkit/target12/usr/local/lib/libnetsnmp.so ~/SBig/Server/lib/
$ make big
```

2.5.4 Compilation des librairies embarquées

Les librairies embarquées sont compilées sur cible en utilisant les scripts mis à disposition dans le code sources. Cf cette page.

2.6 CORBA

2.6.1 Introduction

Le serveur CORBA simule un bus logiciel interfaçable avec les langages C++ et PYTHON. La jointure entre les clients/serveurs se fait via le serveur de nom OMNINAMES.

2.6.2 Deamon OMNINAMES

```
$ nameclt list roche
ctrlSecurity

$ nameclt unbind roche/ctrlSecurity
$ nameclt remove_context roche
```

Ne pas redémarer le deamon (sauf si personne n'utilise CORBA) !

```
# /etc/init.d/omniNames restart
Stopping omniNames                                         [  OK  ]
Starting omniNames                                         [ÉCHOUÉ]
```

Il faut supprimmer ces fichier pour pouvoir redémarer le serveur :

```
/var/omninames:
-rw-r--r-- 1 root root 92678 jan 23  2007 omninames-lpnp90.in2p3.fr.bak
-rw-r--r-- 1 root root 92678 mai 19 14:32 omninames-lpnp90.in2p3.fr.log

/var/omniNames:
-rw-r--r-- 1 omni omni 548 mai 19 14:32 error.log
-rw-r--r-- 1 omni omni  40 mai 19 14:32 omninames-lpnp90.in2p3.fr.bak
-rw-r--r-- 1 omni omni  40 mai 19 14:32 omninames-lpnp90.in2p3.fr.log
```

2.7 RPM

2.7.1 Introduction

Les serveurs ainsi que les cartes controleur embarqués sont basés sur des distribution REDHAT et utilisent donc le gestionnaire de paquet RPM.

Cf ce How-to, ainsi que ce livre.

2.7.2 Commandes en vrac

— information sur le paquet (`apt-cache show`)
```
$ rpm -qpi net-snmp-libs-5.4.2.1-3.fc10.ppc.rpm
```
— lister les fichiers (`dpkg -L`)
```
$ rpm -qlp net-snmp-libs-5.4.2.1-3.fc10.ppc.rpm
```
— chercher quel paquet contient tel fichier (`dpkg -S`)
```
$ rpm -qf /usr/bin/convert
```

2.7.3 Dépecer un RPM

— version bibi
```
# apt-get install alien
# alien net-snmp-libs-5.4.2.1-3.fc10.ppc.rpm
...
```
— version pascal
```
# rpm -i --nodeps --force beecrypt-4.1.2-19.fc11.ppc.rpm
```

2.7.4 réparer YUM

Suite au déplacement de `/var` yum ne répond plus :

```
# yum upgrade
rpmdb: PANIC: fatal region error detected; run recovery
error: db4 error(-30974) from dbenv->open: DB_RUNRECOVERY: Fatal error, run database recovery
error: cannot open Packages index using db3 -  (-30974)
error: cannot open Packages database in /var/lib/rpm
CRITICAL:yum.main:

Error: rpmdb open failed

#  cd /var/lib/rpm
# rm -f __db.00*
# db_verify Packages
# rpm --rebuilddb
# yum update
Loaded plugins: presto, refresh-packagekit
Existing lock /var/run/yum.pid: another copy is running as pid 4631.
Another app is currently holding the yum lock; waiting for it to exit...
  The other application is: yum

# kill -9 4631
# yum update
```

2.8 Pince à sertir

2.8.1 Introduction

Câblage RJ-45. Essayer de se procurer des connecteurs ayant des rails.

2.8.2 Cable droit

Pour un câble "droit" les deux extrémité doivent être placées de cette façon :
— blanc/orange
— orange
— blanc/vert
— bleu
— blanc/bleu
— vert
— blanc/brun
— brun

Vu du côté des broches métalliques, si l'on insert les cables vers la droite, alors l'ordre énoncé sera celui vu de haut en bas, et celà pour les 2 extrémités du câble droit.

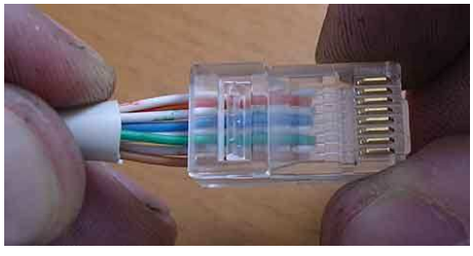


FIGURE 2.1 – cable droit

2.8.3 Cable croisé

Pour un câble "croisé", une extrémité comme ci-dessus, l'autre comme précisé ci-dessous :
— blanc/vert *
— vert *
— blanc/orange *
— bleu
— blanc/bleu
— orange *
— blanc/brun
— brun

2.9 Serveurs de boot

2.9.1 Introduction

Installation de l'environnement de boot des cartes processeur sur un portable UBUNTU, puis sur les 2 pizza boxes qui seront envoyées en Namibie.

2.9.2 Portable Ubuntu

Cette installation permet de simuler l'environnement en vu d'effectuer des développement un châssis en Allemagne.

Environnement

- Serveur SSH et autres outils :
  ```
  # apt-get install ssh tftp
  # apt-get install apt-file
  # apt-file update
  ```
- Nouvelle partition EXT3
  - Dégommage
    ```
    # umount /dev/sda2
    # mke2fs -j /dev/sda2
    ```
  - Fichier */etc/fstab* :
    ```
    /dev/sda2 /media/sda2 ext3 defaults 0 2
    ```
  - Utilisation
    ```
    # mount /dev/sda2
    # mkdir /media/sda2/hess
    ```
- Route vers **n1n9**
  - Congfiguration : fichier */etc/network/interfaces*
    ```
    auto eth0
    iface eth0 inet static
    address 134.158.153.212
    netmask 255.255.248.0
    gateway 134.158.152.1
    up route add -net 192.168.1.0 netmask 255.255.255.0 gw 134.158.152.55
    down route del -net 192.168.1.0 netmask 255.255.255.0 gw 134.158.152.55
    ```
  - Test
    ```
    # /etc/init.d/networking restart
    # route -n
    ```
  - Copie de la partition racine
    ```
    # cd /media/sda2/hess
    # ssh root@192.168.1.9 "cd /opt/CES/shl-3.2.3/eldk && tar -zcf - ppc_74xx_c15"  | tar -zxf -
    # ln -s ppc_74xx_c15 target15
    ```
  - Copie des répertoires personnels
    ```
    # mkdir /media/sda2/hess/home
    # cd !$
    # ssh guevara@192.168.1.9 "cd /mnt/homes/lpnp90/ && tar -zcf - guevara"  | tar -zxf -
    # rsync -e ssh -av guevara@192.168.1.9:/home/guevara/ /media/sda2/hess/home/guevara (maj)
    ```

Serveur TFTP

- Installation
  ```
  # apt-get install tftpd
  # ln -s /media/sda2/hess/tftp /tftpboot
  # mkdir -p /media/sda2/hess/tftp
  # cd !$
  # ln -s /media/sda2/hess/target15/zImage.rio zImage15
  ```
- Configuration : fichier */etc/inetd.conf*
  ```
  tftp dgram udp wait nobody /usr/sbin/tcpd /usr/sbin/in.tftpd /tftpboot
  ```
- Test

```
# tail -f /var/log/daemon.log
                                                remote$ tftp 134.158.153.212
                                                tftp> get /tftpboot/zImage15
                                                Received 1379043 bytes in 0.8 seconds
tftpd: trying to get file: /tftpboot/zImage15
tftpd: read: Connection refused
                                                tftp> ^D
                                                remote$ md5sum zImage15
```

Serveur NFS

— Installation

```
# apt-get install nfs-kernel-server
# cd /media/sda2/hess
# ssh root@192.168.1.9 "cd /opt/CES/shl-3.2.3/eldk && tar -zcf - ppc_74xx_c15"  | tar -zxf -
# ln -s ppc_74xx_c15 target15
# ln -s /media/sda2/hess /opt/CES
# sed -i -e "s!/home/!/opt/ces/home/!" /media/sda2/hess/target15/etc/fstab.local
```

— Configuration : fichier */etc/exports*

```
/media/sda2/hess/target15 *(rw,no_root_squash,no_all_squash)
/media/sda2/hess/home     *(rw,sync,no_root_squash,no_all_squash)
```

— Test

```
# /etc/init.d/nfs-kernel-server restart
# exportfs
# tail -f /var/log/daemon.log
                                remote# mkdir dir
                                remote# mount -t nfs 134.158.153.212:/opt/CES/target15 dir
mountd[5370]: authenticated mount request...
                                remote# umount dir
mountd[5370]: authenticated unmount request...
                                remote# mount -t nfs 134.158.153.212:/opt/CES/home/guevara dir
```

Changer d'IP

Le portable simule les 2 serveurs N1N9 et N1N3.

— Configuration : fichier */etc/network/interfaces*

```
auto eth0
iface eth0 inet static
address 192.168.1.9
netmask 255.255.255.0

auto eth0:2
iface eth0:2 inet static
address 192.168.1.3
netmask 255.255.255.0
```

— Test

```
# /etc/init.d/networking restart
# ifconfig
```

2.9.3 Pizza Boxes

Il s'agit de copier puis migrer l'environnement de développement présent sur N1N9 sur les 2 pizza box N1N40 et N1N41.

Environnement

— */home* des utilisateurs

```
# cd /home
# ln -s /mnt/homes/lpnp90/guevara .
# ln -s /mnt/homes/lpnp90/huppert/ .
# ln -s /mnt/homes/lpnp90/panazol/ .
# ln -s /mnt/homes/lpnp90/roche/ .
# ln -s /mnt/homes/lpnp90/tavernet/ .
# ln -s /mnt/homes/lpnp90/vincentp/ .
```

— / des cartes processeur

```
# cd /data
# mkdir CES
# cd CES
# rsync -e ssh -a root@n1n9:/opt/CES/shl-3.2.3/ /data/CES/shl-3.2.3/
# for i in 2 3 4 5 6 7 ; do ln -s shl-3.2.3/target1$i .; done

# cd /home
# rm bigcamera
# rsync -e ssh -a root@n1n41:/home/bigcamera/ /home/bigcamera/

... ou encore ...
[root@n1n9 ~]# rsync -e ssh -av /opt/CES/shl-3.2.3/eldk/ppc_74xx_c19/ \
                    root@n1n40:/data/CES/shl-3.2.3/eldk/ppc_74xx_c19/
# cd /data/CES/shl-3.2.3
# ln -s eldk/ppc_74xx_c19 target19
# cd ..
# ln -s shl-3.2.3/target19 target19
```

Serveur TFTP

Attention, le réseau hess est protégé par un firerwall sur n1n3. A priori le trafic tftp passe depuis n1n9 vers l'extérieur seulement (pas depuis n1n40 et n1n41).

— Installation

```
# rpm -qa | grep tftp
# yum search tftp
# yum install tftp.i686 tftp-server.i686
  Installing     : 2:xinetd-2.3.14-28.fc12.i686
  Installing     : tftp-0.49-5.fc12.i686
  Installing     : tftp-server-0.49-5.fc12.i686

# rpm -qa | grep tftp
# ln -s /data/tftp /tftpboot
# mkdir /data/tftp
# cd !$
# for i in 12 13 14 15 16 17; do ln -s /data/CES/target$i/zImage.rio zImage$i; done
# yum install tftp.x86_64
```

— Configuration : fichier */etc/xinetd.d/tftp*

Attention aux parametres `server_args` et `disable` :

```
service tftp
{
        socket_type             = dgram
        protocol                = udp
        wait                    = yes
        user                    = root
        server                  = /usr/sbin/in.tftpd
#       server_args             = -s /var/lib/tftpboot
        server_args             = /tftpboot -v
        disable                 = no
        per_source              = 11
        cps                     = 100 2
```

```
        flags                   = IPv4
}
```

— Firewall : il faut ouvrir le port 69.

```
# system-config-firewall
Firewall: [*] Enabled
Customize
[*] TFTP
Close
OK
Yes
```

— Test

```
# /etc/init.d/xinetd stop
# killall in.tftpd
# touch toto.txt
# /usr/sbin/in.tftpd -L
                                                camera15$ tftp n1n40
                                                tftp> get /tftpboot/toto.txt
                                                Received 10 bytes in 0.1 seconds
# tail -f /var/log/messages
in.tftpd[]: tftpd: read(ack): Connection refused
# /etc/init.d/xinetd start

                                                camera15$ tftp n1n40
                                                tftp> get /tftpboot/zImage15

in.tftpd[]: RRQ from 192.168.1.166 filename /tftpboot/zImage15

                                                tftp> ^D
                                                remote$ md5sum zImage15
```

Serveur NFS

— Installation

```
# cd /opt/
# ln -s /data/CES CES
```

— Configuration : fichier */etc/exports*

```
/data/CES/target10 192.168.1.0/255.255.255.0(rw,no_root_squash,no_all_squash)
/data/CES/target11 192.168.1.0/255.255.255.0(rw,no_root_squash,no_all_squash)
/data/CES/target12 192.168.1.0/255.255.255.0(rw,no_root_squash,no_all_squash)
/data/CES/target13 192.168.1.0/255.255.255.0(rw,no_root_squash,no_all_squash)
/data/CES/target14 192.168.1.0/255.255.255.0(rw,no_root_squash,no_all_squash)
/data/CES/target15 192.168.1.0/255.255.255.0(rw,no_root_squash,no_all_squash)
/data/CES/target16 192.168.1.0/255.255.255.0(rw,no_root_squash,no_all_squash)
/data/CES/target17 192.168.1.0/255.255.255.0(rw,no_root_squash,no_all_squash)
/data/CES/target19 192.168.1.0/255.255.255.0(rw,no_root_squash,no_all_squash)
```

— Firerwall : attention, NFSD utilise des ports dynamiques.
Decommenter les ports par default dans le fichier */etc/sysconfig/nfs* :

```
LOCKD_TCPPORT=32803
LOCKD_UDPPORT=32769
MOUNTD_PORT=892
STATD_PORT=662
```

Ajouter ces ports aux regles du firewall dans le fichier */etc/sysconfig/iptables* :

```
# NFS
-A INPUT -m state --state NEW -m tcp -p tcp --dport 111 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 111 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 662 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 662 -j ACCEPT
```

```
-A INPUT -m state --state NEW -m tcp -p tcp --dport 892 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 892 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 2049 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 2049 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 32803 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 32803 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 32769 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 32769 -j ACCEPT
```

(Re)lancer les service :

```
# /sbin/chkconfig nfs on
# /etc/init.d/iptables restart
# /etc/init.d/nfs restart
# /etc/init.d/nfslock restart
```

— Test

```
# /etc/init.d/nfs restart
# exportfs
# tail -f /var/log/messages
                                camera16# mkdir dir
                                camera16# mount -t nfs 192.168.1.40:/opt/CES/target15 dir
mountd[5370]: authenticated mount request...
                                camera16# umount dir
mountd[5370]: authenticated unmount request...

# /sbin/chkconfig | grep nfs
# serviceconf
```

Test et FAQ

— Boot depuis camera15 dans le châssis PCI isolé ok.
— Cross-Compilation ok.
— ne marche pas avec `MTU=9000` mais avec `MTU=1500` sur **n1n41**.
— re-export NFS

```
> > Is it possible to mount a fs via nfs, and then reexport it via nfs?
>
> No.
>
> The protocol doesn't really support it, and the (Linux-kernel)
> implementation definately doesn't support it.
definitelly the kernel implementation cannot manage the loops a re-export
would imply.
>
> I think the user-space nfs server can do it. It has other problems,
> but it might work for you.
yes, you have to run both rpc.mountd and rpc.nfsd with the -r option.I
used this one year ago, and it was really stable with 10 nfs clients
(physicaly the partition was on an AIX server, mounted on a linux server
and then re-exported on many linux clients), but the physical FS status
could not be really coherent in front of what the nfs re-export server
thinks it should be.
```

— Comment **n1n4[12]** montent-ils les homes ?
Il y a des pages jaunes sur **n1n3**. Cf le fichier */etc/ntp.conf* :

```
server 192.168.1.3 dynamic
```

2.10 Installation des postes utilisateurs

2.10.1 Introduction

Machine concernées par cette procédure :

```
192.168.1.15    n1n15.in2p3.fr  n1n15  # fc12 (sous-sol)
192.168.1.31    n1n30.in2p3.fr  n1n30  # fc12 (salle camera, Julien Bolmon)
192.168.1.31    n1n31.in2p3.fr  n1n31  # fc12 (salle camera, Julien Bolmon)
192.168.1.40    n1n40.in2p3.fr  n1n40  # fc12 pizza 64 (sous-sol)
192.168.1.41    n1n41.in2p3.fr  n1n41  # fc12 pizza 64 Namibie (sous-sol)
192.168.1.42    n1n42.in2p3.fr  n1n42  # fc9? PC double écran (bureau JP)
192.168.1.43    n1n42.in2p3.fr  n1n43  # fc12 PC double écran Namibie (sous-sol)
```

2.10.2 Préparation de la mise à niveau

Avant de basculer une machine sous fedora 12, noter
- la configuration réseau (adresse IP, gateway, masque)
- le partitionnement
  ```
  # lvm pvdisplay /dev/sda5
  # lvm vgdisplay /dev/VolGroup00
  # lvm lvdisplay /dev/VolGroup00/LogVol00
  ```

2.10.3 Installation Fedora 12

Graver le DVD Fedora 12. On peut aussi utiliser CD d'installation réseau , mais on aura besoin du réseau lors de l'installation. **Rq :** penser à télécharger aussi le CD pour l'architecture X86_64.

Effectuer une installation standard (*install system with basic video driver*), en faisant attention aux points suivants :
- *custum partitions* :
  - réutiliser en les formatant les partitions `/`, `/usr`, `swap`, `/var`, `/home` et eventuelement `/usr/local` préalablement utilisées.
  - réutiliser **sans la formater** l'eventuelle partition `/data` préalablement utilisée.
- horloge système réglée sur UTC
- (reboot sans le CD)
- créer un compte local `lpnhe`
- bouton *use network login* facultatif (cf ci dessous les paramètres de `system-config-authentification`)
- synchronisation temporelle facultative (cf ci-dessous les paramètres de `system-config-date`)

Une fois Fedora 12 installée,
- modifier le *gid* du groupe `lpnhe`
  - dans le fichier */etc/group* :
    `lpnhe:x:200:`
  - et dans le fichier */etc/passwd* :
    `lpnhe:x:500:200:Administrateur:/home/lpnhe:/bin/bash`
- faire une mise à jours :
  ```
  # yum upgrade
  # yum install -y csh zsh emacs
  ```
- Désactiver **SELinux** (sinon on a du mal à se loguer via NIS, c'est-à-dire que l'on arrive sur la partition racine au lieu du répertoire personnel). Fichier */etc/selinux/config* :
  ```
  SELINUX=permissive
  SELINUXTYPE=targeted
  ```
- Vous pouvez utiliser `setenforce 0` pour activer le mode permissif mais le plus prudent est de rebooter.
  ```
  # reboot
  ```

2.10.4 Synchronisation temporelle

lpnp90 est synchronisée sur le routeur du labo. Les machines doivent être synchronisée sur lpnp90 :

```
# yum install -y ntp system-config-date
# system-config-date
```

— Cocher *Synchronizer la date et l'heure sur le réseau*.
— Supprimer les serveurs proposés et ajouter 192.168.1.3 comme nouveau serveur de temps.
— Dans les options, cocher Synchroniser l'horloge système avant de lancer le service.

2.10.5 Authentification

NIS

NIS doit être installé. Si ce n'est pas le cas :

```
# yum install -y ypbind authconfig-gtk
```

Lancer la configuration d'authentification :

```
# system-config-authentication
```

Choisir :
— authentification : NIS
— réseau : **hess**
— serveur : **192.168.1.3**

Editer le fichier */etc/nsswitch.conf* pour faire en sorte que la liste des machines soit lue à travers NIS, ainsi que les montages réseau. Les lignes importantes sont :

```
passwd:     files nis
shadow:     files nis
group:      files nis
hosts:      files nis dns
automount:  files nis
services:   files nis
```

Pour vérifier que la liste des machines est bien accessible, vérifier que les machines du réseau hess sont accessibles :

```
# ypcat hosts
# ping n1n3
```

Si ça ne marche pas, il faut ajouter la route suivante via le fichier */etc/sysconfig/network-scripts/route-eth0* :

```
GATEWAY0=134.158.152.55
NETMASK0=255.255.255.0
ADDRESS0=192.168.1.0
```

autofs

Installer autofs, et effacer les fichiers de configuration autofs (car on utilise NIS) :

```
# yum install -y autofs
# rm -f /etc/auto.*
```

Démarrer ensuite le démon autofs, et le configurer pour un démarrage automatique :

```
# /sbin/chkconfig autofs on
# /etc/rc.d/init.d/autofs start
```

Vérification du fonctionnement autofs (création des répertoires et montage réseau) :

```
# ls /mnt/homes/lpnp90
# ls /mnt/misc/data
```

On doit avoir la liste des utilisateurs. Le répertoire /mnt/misc/data doit aussi être accessible. Si nécessaire, ajouter les machines dans /etc/exports sur lpnp90 et recharger NFS (`/etc/rc.d/init.d/nfs reload`)

Montage des homes :

```
# cd /home
# ln -s /mnt/homes/lpnp90/* .
```

Les homes et les comptes devraient maintenant être accessibles.

2.10.6 Installation des softs HESS

Installer le repository *hess* pour yum

Télecharger le fichier http ://lpnp90.in2p3.fr/~hess/Software/For_Fedora12/hess-release-9-5.noarch.rpm puis l'installer.

```
# rpm -i hess-release-9-5.noarch.rpm
```

Installer le repository rpmfusion

RPM Fusion provides software that the Fedora Project or Red Hat doesn't want to ship. That software is provided as precompiled RPMs for all current Fedora versions and Red Hat Enterprise Linux 5 ; you can use the RPM Fusion repositories with tools like yum and PackageKit.

RPM Fusion is a merger of Dribble, Freshrpms, and Livna ; our goal is to simplify end-user experience by grouping as much add-on software as possible in a single location.

```
# rpm -i http://download1.rpmfusion.org/free/fedora/rpmfusion-free-release-stable.noarch.rpm
# rpm -i http://download1.rpmfusion.org/nonfree/fedora/rpmfusion-nonfree-release-stable.noarch.rpm
```

Installer CORBA

Les RPM CORBA sont dans le repository `hess` (cf ci-dessus).

```
# yum install -y omniORB omniORB-devel omniORBpy omniORBpy-devel omniORB-utils
```

Configurer l'accès au serveur CORBA : éditer le fichier */etc/omniORB.cfg*. La ligne importante est :

```
InitRef = NameService=corbaname::n1n3
```

Dans le cas où la machine possède plusieurs interfaces réseau, il faut préciser laquelle on utilise :

```
# cat /etc/profile.d/omniORB.sh
export OMNIORB_USEHOSTNAME=192.168.7.40

# /etc/profile.d/omniORB.csh
setenv OMNIORB_USEHOSTNAME 192.168.7.40
```

Tester le bus CORBA :

```
# nameclt list
hess/
```

Penser à désactiver le firerwall :

```
# system-config-firewall
```

Installer DASH

```
# yum install -y hess-dash hess-dash-devel hess-dash-python hess-dash-sash hess-dash-sash-devel
```

Si le cache de yum est plein le vider préalablement avec `# yum clean all`.

Installer les controlleurs

Chacun des controlleur a son propre module. Par exemple :

Controlleur camera (HESS-I)	: `# yum install -y hess-camera`
CameraReader	: `# yum install -y hess-camerareader`
Bras X-Y	: `# yum install -y hess-owis`
Instruments gpib	: `# yum install -y hess-gpib_instruments`
Testbench (analyse de données)	: `# yum install -y hess-testbench-gui hess-testbench`
Module de calibration	: `# yum install -y hess-onlinecalibrator`

```
# yum install -y hess-camera hess-camerareader hess-owis hess-gpib_instruments \
                 hess-testbench-gui hess-testbench hess-onlinecalibrator
```

Installer LIBNETSNMP pour compiler l'outils power

```
# yum install net-snmp-libs
```

Installer le contrôleur camera2

Installer le code depuis le module CVS

```
$ mkdir hess
$ cvs -d :pserver:hess@hess01.physik.hu-berlin.de:/cvs login
$ cvs -d :pserver:hess@hess01.physik.hu-berlin.de:/cvs co camera2
# yum install -y hess-onlinestarcontrol
# yum install -y hess-onlinesound
```

Modifier la configuration des controleurs (géographie et path) dans le fichier *hess/camera2/python/ConfigWindow.py*.

Installer la fonte TIMES

Les fontes utilisées sont celles du serveur X11 :

```
# yum install xorg-x11-fonts-*
# xlsfonts | grep '\-times-bold-r-normal-.*-180-'
```

Rq : Pourtant la fonte TIMES fait a priori parti des font Microsoft.

```
# INUTILE

# yum install xfs
# rpm -ivh http://dl.atrpms.net/all/chkfontpath-1.10.1-2.fc12.i686.rpm      // lent
# yum install rpm-build cabextract ttmkfdir wget
# wget http://corefonts.sourceforge.net/msttcorefonts-2.0-1.spec
# rpmbuild -ba msttcorefonts-2.0-1.spec
```

Configurer le controleur camera2

— fichier */.dashrc* :

```
DBSERVER=n1n3.in2p3.fr
DBUSERNAME=hessdaq
DBPASSWD=
DBNAME=HESS_DAQ_Paris_TestBench
DBPORT=3306
RUN_PATH=/mnt/misc/data/camera_20
CALIBDB=CALIBRATION
DAQCLUSTER=hess-paris
CALIBPASSWD=hessdaq
CALIBUSERNAME=hessdaq
CALIB_PATH=/mnt/misc/data/Calibration/
SAMTEST=SAM_Paris
DRAWERTEST=DRAWER_Paris1
TESTBENCH_RESULTS_PATH=/mnt/misc/data/TestBench_Results_Camera20
HESSDOCVIEW=galeon
```

— fichier */.dbtoolsrc* :

```
[hessdaq]
host=n1n3.in2p3.fr
user=hessdaq
password=
database=HESS_DAQ_Paris_TestBench
```

— répertoire */mnt/misc/data/* :

```
$ mkdir /mnt/misc/data/camera_20
$ ln -s hess/camera2/start-camera88 .
$ touch ~/.alias
```

— fichier */.zshenv* :

```
source ~/.zprofile
```

Ce fichier permet d'exporter les variables d'environnement lorsqu'une commande est exécuté via SSH.

```
$ ssh n1n15 env | grep HESS
HESSROOT=/usr/local/hess
HESSUSER=/home/camera2/hess
```

Installer omniNames

Il s'agit du serveur du bus CORBA installé sur N1N3.

```
# yum install omniORB-servers omniORB-bootscripts
```

Remarque : Il faut lancer le serveur manuellement pour voir qu'un répertoire manque (*/var/omninames/* je crois). Il suffit de le créer (et peut-être aussi de modifier les permissions dessus).

```
# su - omni
$ omniNames -ORBtracelevel 25
```

Ensuite, il faut vérifier les fichier */etc/hosts* :

```
127.0.0.1   localhost localhost.localdomain
```

et le fichier */etc/omniORB.conf* :

```
InitRef = NameService=corbaname::n1n15.in2p3.fr
```

2.11 Installation des switchs utilisateurs

2.11.1 Introduction

Configuration des switch sur le Vlan du labo.

2.11.2 Netgear GS108

— Appuyer 7 secondes sur le bouton “reset”
— Connecter le switch sur le réseau `192.168.0.0/255.255.255.0`
— Ouvrir la page WEB http ://192.168.0.239 avec un navigateur
— Entrer le mot de passe `password`

Changer le mot de passe

— Aller à `Security >>> Password`
— Changer le mot de passe pour le smac réduit.
— Valider

Changer le nom du switch

— Aller à `System >>> Info`
— Changer le hostname pour `lpnsw87` par exemple
— Valider

Configurer le Vlan

Créer le Vlan 199

— Aller à `Switching >>> Vlan >>> Vlan configuration`
— Utiliser la norme 802.1Q
— Créer le Vlan 199
— Valider

Affecter un Vlan à chacun des ports du switch (1)

— Aller à `Switching >>> Vlan >>> Membership`
— Choisir le Vlan 199
— Positionner chaque port à `U`
— Valider

Affecter un Vlan à chacun des ports du switch (2)

— Aller à `Switching >>> Vlan >>> PVID Configuration`
— Choisir le Vlan 199 pour chacun des ports
— Valider

Changer l’IP du switch

— Aller à `System >>> IP`
— `address: 134.158.152.xxx`
— `netmask: 255.255.252.0`
— `gateway: 134.158.152.1`
— `Management Vlan: 199`

— Valider

2.11.3 Allied AT-800GS/48

Effacer la configuration actuelle :

```
User Name:manager
Password:***

lpnswall03> enable
Password:***

lpnswall03# show run
...
lpnswall03# dir
lpnswall03# delete startup-config
lpnswall03# reload

User Name:manager
Password:friend

console# show run
Empty configuration
```

Troisième partie

Annexes

— Les milestones ("jalons") avec les deadlines

Démon LED avec interface CORBA	fait
Connectique cPCI : co-habitation maitres pci, bridges et gigabit	fait
Implémentation du serveur Bunny	proto pour le LAPP
Implémentation du serveur Big	en cours
Implémentation du serveur Zora	JF
Implémentation du serveur Emilie	JF
Basculer le serveur de boot	fait
Controlleur CORBA du chassis sécurité	proto pour le LAPP
Banc de test pour le local module	fait
Mise à jour des postes et serveurs sous Fedora7	à faire

— Liste de vos taches à faire (avec échéance éventuelle)

Disposer une carte controlleur au delà du 1er bridge PCI sur le châssis 21 slots	en attente
Uniformiser les automates avec le “Array Manager”	en cours
Basculer le serveur de boot	fait
Code C de la gestion des alims et slow contôlre	en cours
Accusé de réception sur mise à jour des commandes Drawers	en cours
Intérogation du GPS via Big	en cours
Watchdog sur Zora	en attente
GPIO via le serveur de console	en attente

— Liste de vos taches faites

Démon LED
Cross-compilateur ARM sous debian Serveur de socket sur carte ARM Serveur CORBA en PYTHON + GUI.
Connectique cPCI
Mapper les cartes PCI depuis le second slot du chassis Flashage EPROM de la carte ethernet gigabit et mesure des débits Cross-compilateur PPC Franchir le 2ème bridge PCI sur le châssis 21 slots Banc de test pour le local module
Prototype pour le LAPP
Serveur Bunny Controlleur sécurité CORBA en C++ + GUI

— Difficultés éventuelles rencontrées

Localisation des picots utilisables pour les GPIOGPIO
Adresses IP clientes inutilisables pour les montages NFS
Travail supplémentaire pour adapter le code du LAPP
GPS non interrogeable si mal initialisé (=> antenne ok).

— Les formations

vaccins hépatite A et typhoïde	fait le 7/4/9

— Vos prochaines réunions/missions du projet

Le mardi à 10H30 tous les 15 jours.

Bibliographie

[1] *SYSTÈME DE CALIBRAGE CAMÉRA*. SYSTEME DE LED [LPNHE-Paris].

Annexe A

Index

Index