

Cross-Compile

Table des matières

1	Introduction	1
2	Test des outils ces sous Debian	1
2.1	Installation des outils et d'ELDK	1
2.2	Cross-compilation du noyau LINUX	1
3	Cross-compilation sur n1n9	2
3.1	test de Cross-compilation du noyau LINUX	2
3.2	Installation d'ELDK	2
3.3	Configuration du système d'ELDK	2
3.4	cross-compilation du noyau LINUX	3
4	Cross-compilation sur n1n40	3
5	Cross-compilation sur Ubuntu	3
5.1	Installation d'ELDK	3
5.2	Cross compilation de BIG	4
6	Compilation sur cible (carte contrôleur)	4
6.1	Ajout de la bibliothèque NET-SNMP	4
6.2	compilation du noyau LINUX	5
7	Debugage	5
7.1	Console	5
7.2	Emacs	5
7.3	FAQ	6

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.

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.

2.1 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.

2.2 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 Cross-compilation sur n1n9

3.1 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
```

3.2 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 28/05/2009
^D
```

```
# cp -Rp ppc_74xx_20090528 ppc_74xx_20090528_c12
```

3.3 Configuration du système d'Eldk

Attention : ici nous faisons abstraction des liens symboliques :

- Pensez à utiliser `# tar` ou `cp -Rp` pour dupliquer 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).

3.4 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-3.2.3"
# 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
```

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.

5 Cross-compilation sur Ubuntu

5.1 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-*
```

5.2 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
```

6 Compilation sur cible (carte contrôleur)

6.1 Ajout de la bibliothèque Net-Snmp

Il s'agit d'installation sur une distribution LINUX FROM SCRATCH et par conséquent, bien que l'outil RPM soit disponible, aucun paquet n'est installé. Par ailleurs l'installation d'un meta-gestionnaire de paquet (APT) nécessite 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 disposition. 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 contrôleur.

- *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
```

6.2 compilation du noyau Linux

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

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

7 Debuggage

Sisi c'est possible.

7.1 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 : cela marche aussi avec EMACS :

7.2 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
```

7.3 FAQ

- Parfois on perd le code source (à priori) à cause de la compilation réursive. 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:$cd:$cd
```

- 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
```