LFS : Lab 2 - Construire un filesystem initram depuis une chaîne d'outils de compilation croisée

Objet	construction d'une toolchain dans un initram
Niveau requis	débutant, avisé
Features	
Débutant, à savoir	lfs-lab1-clfs-toolchain
Suivi	:INPROGRESS:

Description

Ce lab décrit la construction d'un système embarqué depuis la toolchain construite dans le LAB1.

Préparation du système hôte

Ajout de l'utilisateur LFS

Construire un système from scratch en tant qu'utilisateur root est dangereux, car commettre une seule erreur peut endommager ou détruire le système hôte. Il est préférable de construire les packages en tant qu'utilisateur non privilégié. Pour faciliter la configuration d'un environnement de travail, créer un nouvel utilisateur appelé `lfs` en tant que membre du groupe `clfs` (créé dans la LAB1) et utiliser cet utilisateur lors de la procédure d'installation:

```
useradd -s /bin/bash -g clfs -m -k /dev/null lfs
echo -e "clfspassword\nclfspassword" | passwd lfs
usermod -aG wheel lfs
su - lfs
```

Les fichiers suivants sont nécessaires pour s'assurer que l'environnement correct est lu pour chacune des façons dont le shell peut être appelé. Le fichier `~/.bash_profile` est lu quand le shell est invoqué comme shell interactif de connexion. Le fichier `~/.bashrc` est lu quand le shell est invoqué comme shell interactif sans fonction de connexion.

```
cat > ~/.bashrc << "EOF"
set +h
umask 022
CLFS=/var/lfs/clfs
LC_ALL=POSIX
PATH=${CLFS}/toolchain/bin:/bin:/usr/bin
export CLFS LC_ALL PATH
EOF
```

```
cat > ~/.bash_profile << "EOF"
exec env -i HOME=${HOME} TERM=${TERM} PS1='\u:\w\$ ' /bin/bash
EOF
```

```
source ~/.bash_profile
unset CFLAGS
echo unset CFLAGS >> ~/.bashrc
```

Définition des variables de construction

```
ARCH="x86"
CPU="i686"
HOST=$(uname -m)-$OSTYPE
export TARGET=i686-linux-musl
export HOST=$MACHTYPE
export TCDIR=${CLFS}/toolchain
export CLFS="${CLFS}"
```

Construction de l'environnement

```
mkdir -pv ${CLFS}/{build,src,targetfs}
```

Téléchargement des paquets

Télécharger les packages dans le répertoire ${CLFS}/download.

Confection de la liste des paquets

Les packages source suivants on été téléchargé pour la construction de la chaîne d'outils dans le LAB1:

- Linux - il s'agit d'un noyau très populaire fonctionnant sur notre système cible, il nous faut au moins les en-têtes pour construire la bibliothèque standard C ().
- Musl - Une minuscule implémentation de la bibliothèque standard C.
- Binutils - contient l'assembleur GNU, l'éditeur de liens et divers outils permettant de travailler avec des fichiers exécutables.
- GCC - contient les compilateurs pour C et d'autres langages.
- MPFR - bibliothèque à virgule flottante à multiples précisions.
- GMP - bibliothèque arithmétique à multiples précisions.
- MPC - bibliothèque de précisions multiples pour les nombres complexes.
- ISL - bibliothèque mathématique nécessaire pour optimiser les boucles.
- CLOOG - bibliothèque de générateur de code.

```
cat >> ${CLFS}/download/dl-src.list << EOF
https://zlib.net/zlib-1.2.11.tar.xz
ftp://ftp.astron.com/pub/file/file-5.36.tar.gz
http://ftp.gnu.org/gnu/readline/readline-8.0.tar.gz
http://ftp.gnu.org/gnu/m4/m4-1.4.18.tar.xz
https://github.com/gavinhoward/bc/archive/2.1.3/bc-2.1.3.tar.gz
http://anduin.linuxfromscratch.org/LFS/bzip2-1.0.6.tar.gz
http://ftp.gnu.org/gnu/ncurses/ncurses-6.1.tar.gz
http://ftp.gnu.org/gnu/sed/sed-4.7.tar.xz
http://anduin.linuxfromscratch.org/LFS/iana-etc-2.30.tar.bz2
https://github.com/westes/flex/releases/download/v2.6.4/flex-2.6.4.tar.gz
http://ftp.gnu.org/gnu/grep/grep-3.3.tar.xz
http://ftp.gnu.org/gnu/bash/bash-5.0.tar.gz
http://ftp.gnu.org/gnu/gperf/gperf-3.1.tar.gz
http://ftp.gnu.org/gnu/automake/automake-1.16.1.tar.xz
http://ftp.gnu.org/gnu/gawk/gawk-4.2.1.tar.xz
http://www.greenwoodsoftware.com/less/less-530.tar.gz
http://ftp.gnu.org/gnu/make/make-4.2.1.tar.bz2
http://ftp.gnu.org/gnu/patch/patch-2.7.6.tar.xz
ftp://ftp.vim.org/pub/vim/unix/vim-8.1.tar.bz2
http://matt.ucc.asn.au/dropbear/releases/dropbear-2019.78.tar.bz2
https://busybox.net/downloads/busybox-1.31.0.tar.bz2
https://downloads.sourceforge.net/nfs/nfs-utils-2.3.3.tar.xz
ftp://ftp.vim.org/pub/vim/runtime/spell/fr.utf-8.spl
ftp://ftp.vim.org/pub/vim/runtime/spell/fr.utf-8.sug
EOF
```

Téléchargement des paquets et des correctifs

```
wget -i ${CLFS}/download/dl-src.list -P ${CLFS}/download
```

Création du système cible

Définition des variables de la ToolChain

```
cp -a ~/.bashrc ~/.bashrc-bkp
echo export CC=\""${TARGET}-gcc --sysroot=${CLFS}/targetfs\"" >> ~/.bashrc
echo export CXX=\""${TARGET}-g++ --sysroot=${CLFS}/targetfs\"" >> ~/.bashrc
echo export AR=\""${TARGET}-ar\"" >> ~/.bashrc
echo export AS=\""${TARGET}-as\"" >> ~/.bashrc
echo export LD=\""${TARGET}-ld --sysroot=${CLFS}/targetfs\"" >> ~/.bashrc
echo export RANLIB=\""${TARGET}-ranlib\"" >> ~/.bashrc
echo export READELF=\""${TARGET}-readelf\"" >> ~/.bashrc
echo export STRIP=\""${TARGET}-strip\"" >> ~/.bashrc
source ~/.bashrc
```

Création des répertoires

```
mkdir -pv ${CLFS}/targetfs/{bin,boot,dev,etc,home,lib/{firmware,modules}}
mkdir -pv ${CLFS}/targetfs/{mnt,opt,proc,sbin,srv,sys}
mkdir -pv ${CLFS}/targetfs/var/{cache,lib,local,lock,log,opt,run,spool}
install -dv -m 0750 ${CLFS}/targetfs/root
install -dv -m 1777 ${CLFS}/targetfs/{var/,}tmp
mkdir -pv ${CLFS}/targetfs/usr/{,local/}{bin,include,lib,sbin,share,src}
```

Création des fichiers passwd, group et lastlog

Le fichier `/etc/passwd` est un fichier texte, chaque enregistrement décrivant un compte d'utilisateur. Chaque enregistrement se compose de sept champs séparés par un deux-points. L'ordre des champs dans l'enregistrement est généralement sans importance. Les champs, de gauche à droite, sont :

- le nom de l'utilisateur (login name). C'est la chaîne de caractères que l'utilisateur entre lorsqu'il se connecte au système. Le nom d'utilisateur doit être unique dans le système.
- le mot de passe d'un utilisateur. Dans la plupart des usages modernes, ce champ contient “x” (ou “*”, ou un autre indicateur) et les informations de mot de passe étant stockées dans un fichier de mots de passe séparé. Sur les systèmes Linux, “*” empêche les connexions d'un compte tout en conservant son nom d'utilisateur, alors que “*NP*” indique d'utiliser un serveur NIS pour obtenir le mot de passe2. Avant le masquage du mot de passe (c'est-à-dire dans les premières implantations de Unix), ce champ contenait un hachage cryptographique du mot de passe de l'utilisateur (en combinaison avec un sel).
- l'identifiant d'utilisateur. Un nombre utilisé par le système d'exploitation à des fins internes. Il n'est pas nécessairement unique.
- l'identifiant de groupe. Un nombre qui identifie le groupe principal de l'utilisateur. Tous les fichiers créés par cet utilisateur peuvent être initialement accessibles à ce groupe.
- le champ Gecos. Un commentaire qui décrit la personne ou le compte. Généralement, il s'agit d'un ensemble de valeurs séparées par des virgules, fournissant le nom complet de l'utilisateur et ses coordonnées.
- le chemin vers le répertoire personnel de l'utilisateur.
- le programme qui est lancé chaque fois que l'utilisateur se connecte au système. Pour un utilisateur interactif, il s'agit généralement d'une interface en ligne de commande.

```
cat > ${CLFS}/targetfs/etc/passwd << "EOF"
root::0:0:root:/root:/bin/ash
adm:x:3:16:adm:/var/adm:/bin/false
nobody:x:65534:65534:nobody:/:/bin/false
EOF
```

Le fichier `/etc/group` est un fichier ASCII qui définit les groupes auxquels appartient chaque utilisateur. Il y a une ligne par groupe, et chaque ligne a le format `nom_du_groupe:mot_de_passe:GID:liste_utilisateurs`

```
cat > ${CLFS}/targetfs/etc/group << "EOF"
```

```
root:x:0:
bin:x:1:
sys:x:2:
kmem:x:3:
tty:x:4:
tape:x:5:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
dialout:x:10:
audio:x:11:
video:x:12:
utmp:x:13:
usb:x:14:
cdrom:x:15:
adm:x:16:root,adm,daemon
console:x:17:
users:x:100:
nogroup:x:65533:
nobody:x:65534:
EOF
```

Le fichier `/var/log/lastlog` est le journal des connexions, il est dans un format binaire.

```
touch ${CLFS}/targetfs/var/log/lastlog
chmod -v 664 ${CLFS}/targetfs/var/log/lastlog
ln -svf ../proc/mounts ${CLFS}/targetfs/etc/mtab
```

Installation des en-têtes du noyau

(TARGETFS) Récupération des en têtes du noyau en 53.042 seconds ...

Pour localiser tous les fichiers de sortie dans un répertoire séparé, les deux syntaxes supportées doivent être utilisées: définir la variable d'environnement KBUILD_OUTPUT pour qu'elle pointe vers le répertoire où les fichiers de sortie doivent être placés, `export KBUILD_OUTPUT=dir/to/store/output/files/`, **ET** utiliser l'option O= dans l'intruction make, `make O=$KBUILD_OUTPUT` (le répertoire de travail doit être la racine src du noyau).

```
TIMEFORMAT='(TARGETFS) Récupération des en têtes du noyau en %R seconds ...'
time {
export KBUILD_OUTPUT="$CLFS/build/linux" &&
mkdir -pv "$KBUILD_OUTPUT" &&
tar xf ${CLFS}/download/linux-4.20.12.tar.xz -C ${CLFS}/src &&
cd ${CLFS}/src/linux-4.20.12 &&
make mrproper &&
```

```
make O="$KBUILD_OUTPUT" ARCH="$ARCH" headers_check &&
find ${CLFS}/build/linux/include \( -name .install -o -name ..install.cmd \)
-delete
cp -rv ${CLFS}/build/linux/include/* ${CLFS}/targetfs/include
cd "${CLFS}"
}
```

Compilation de cross binutils

(TARGETFS) Compilation de cross binutils en 257.198 seconds ....

Le paquet Binutils contient un éditeur de liens, un assembleur et d'autres outils pour gérer des fichiers objets.

La documentation de Binutils recommande de construire Binutils dans un répertoire de construction dédié :

```
TIMEFORMAT='(TARGETFS) Compilation de cross binutils en %R seconds ...'
time {
tar xjf ${CLFS}/download/binutils-2.27.tar.bz2 -C ${CLFS}/src &&
mkdir -p "${CLFS}/build/binutils-t" &&
cd "${CLFS}/build/binutils-t" &&
srcdir="${CLFS}/src/binutils-2.27" &&
$srcdir/configure --prefix=/usr \
    --host=$TARGET              \
    --target=$TARGET            \
    --disable-nls               \
    --disable-werror            \
    --with-lib-path=$TCDIR/lib  \
    --with-sysroot &&
make tooldir=/usr &&
make DESTDIR=${CLFS}/targetfs tooldir=/usr install &&
cd "${CLFS}"
}
```

Installation de musl

(TARGETFS) Compilation de la librairie standard C en 116.504 seconds ...

Le paquet musl contient la bibliothèque principale C. Cette bibliothèque fournit les routines de base pour l'allocation de mémoire, la recherche de répertoires, l'ouverture et la fermeture de fichiers, la lecture et l'écriture de fichiers, la gestion de chaînes, la correspondance de modèles, l'arithmétique, etc.

```
TIMEFORMAT='(TARGETFS) Compilation de la librairie standard C en %R seconds
...'
time {
tar xzf ${CLFS}/download/musl-1.1.23.tar.gz -C ${CLFS}/src &&
mkdir -p "${CLFS}/build/musl-tfs" &&
cd "${CLFS}/build/musl-tfs" &&
srcdir="${CLFS}/src/musl-1.1.23" &&
$srcdir/configure \
  CROSS_COMPILE=${TARGET}- \
  --prefix=/ \
  --target=${TARGET} &&
make &&
make DESTDIR=${CLFS}/targetfs install-libs &&
cd "${CLFS}"
}
```

Installation de GCC

(TARGETFS) Compilation du compilateur GCC en 3840.630 seconds ...

Le paquet GCC contient la collection de compilateurs GNU, qui inclut les compilateurs C et C++.

La documentation de GCC recommande de construire GCC dans un répertoire de construction dédié :

```
TIMEFORMAT='(TARGETFS) Compilation du compilateur GCC en %R seconds ...'
time {
tar -xf ${CLFS}/download/gcc-8.2.0.tar.xz -C ${CLFS}/src &&
tar -xf ${CLFS}/download/mpfr-4.0.2.tar.xz -C ${CLFS}/src &&
tar -xf ${CLFS}/download/gmp-6.1.2.tar.xz -C ${CLFS}/src &&
tar -xzf ${CLFS}/download/mpc-1.1.0.tar.gz -C ${CLFS}/src &&
tar -xf ${CLFS}/download/isl-0.16.1.tar.xz  -C ${CLFS}/src &&
tar -xzf ${CLFS}/download/cloog-0.18.1.tar.gz -C ${CLFS}/src &&
cd ${CLFS}/src/gcc-8.2.0 &&
ln -svf "${CLFS}/src/gmp-6.1.2" "gmp" &&
ln -svf "${CLFS}/src/mpc-1.1.0" "mpc" &&
ln -svf "${CLFS}/src/mpfr-4.0.2" "mpfr" &&
ln -svf "${CLFS}/src/cloog-0.18.1" "cloog" &&
ln -svf "${CLFS}/src/isl-0.16.1" "isl" &&
mkdir -p "${CLFS}/build/gcc-t" &&
cd "${CLFS}/build/gcc-t" &&
srcdir="${CLFS}/src/gcc-8.2.0" &&
$srcdir/configure --prefix="/usr" --target="$TARGET" \
                  --host="$TARGET" \
                  --with-sysroot="$CLFS/targetfs" \
                  --disable-nls --enable-languages=c,c++ \
                  --enable-c99 --enable-long-long \
                  --disable-libmudflap --disable-multilib \
```

```
                  --disable-libmpx --disable-libssp --disable-libsanitizer \
                  --with-arch="$CPU" \
                  --with-native-system-header-dir="/include" &&
make &&
make DESTDIR=${CLFS}/targetfs install &&
rm -rf ${CLFS}/targetfs/usr/lib/gcc/$TARGET/8.2.0/include-fixed/bits/ &&
sudo chown -v -R root:root \
    ${CLFS}/targetfs/usr/lib/gcc/$TARGET/8.2.0/include{,-fixed} &&
cd ${CLFS}/targetfs/lib &&
ln -svf ../usr/bin/cpp . &&
cd ${CLFS}/targetfs/usr/bin &&
ln -svf gcc cc &&
install -v -dm755 ${CLFS}/targetfs/usr/lib/bfd-plugins &&
${CLFS}/targetfs/usr/lib/bfd-plugins/ &&
ln -svf ../../libexec/gcc/$TARGET/8.2.0/liblto_plugin.so . &&
cd "${CLFS}"
}
```

Installation de Zlib

(TARGETFS) Compilation de zlib-1.2.11 en 12.711 seconds ...

Le paquet Zlib contient des routines de compression et décompression utilisées par quelques programmes.

```
TIMEFORMAT='(TARGETFS) Compilation de zlib-1.2.11 en %R seconds ...'
time {
tar xf ${CLFS}/download/zlib-1.2.11.tar.xz -C ${CLFS}/src &&
cd ${CLFS}/src/zlib-1.2.11 &&
./configure --prefix=/usr &&
make &&
make prefix=${CLFS}/targetfs install &&
cd ${CLFS}/targetfs/usr/lib &&
ln -sfv libz.so.1.2.11 libz.so.1 &&
cd "${CLFS}"
}
```

Installation de M4

(TARGETFS) Compilation de m4-1.4.18 en 41.867 seconds ...

Le paquet M4 contient un processeur de macros.

```
TIMEFORMAT='(TARGETFS) Compilation de m4-1.4.18 en %R seconds ...'
time {
tar xf ${CLFS}/download/m4-1.4.18.tar.xz -C ${CLFS}/src&&
cd ${CLFS}/src/m4-1.4.18 &&
./configure --prefix=/usr    \
            --target=$TARGET \
            --host=$TARGET &&
make &&
make DESTDIR=${CLFS}/targetfs install &&
cd "${CLFS}"
}
```

Installation de Ncurses

(TARGETFS) Compilation de ncurses-6.1 en 86.815 seconds ...

Le paquet Ncurses contient les bibliothèques de gestion des écrans type caractère, indépendant des terminaux.

```
TIMEFORMAT='(TARGETFS) Compilation de ncurses-6.1 en %R seconds ...'
time {
tar xzf ${CLFS}/download/ncurses-6.1.tar.gz -C ${CLFS}/src&&
cd ${CLFS}/src/ncurses-6.1 &&
./configure --prefix=/usr    \
            --target=$TARGET \
            --host=$TARGET   \
            --without-debug  \
            --with-shared \
            --without-normal \
            --enable-widec &&
make &&
make DESTDIR=${CLFS}/targetfs install &&
mv -v ${CLFS}/targetfs/usr/lib/libncursesw.so.6* ${CLFS}/targetfs/lib &&
cd ${CLFS}/targetfs/usr/lib/ &&
ln -sfv ../../lib/$(readlink ${CLFS}/targetfs/usr/lib/libncursesw.so)
libncursesw.so &&
rm -vf                     ${CLFS}/targetfs/usr/lib/libcursesw.so &&
echo "INPUT(-lncursesw)" > ${CLFS}/targetfs/usr/lib/libcursesw.so &&
cd ${CLFS}/targetfs/usr/lib/ &&
ln -sfv libncurses.so libcurses.so &&
cd "${CLFS}"
}
```

Installation de File

Le paquet File contient un outil pour déterminer le type d'un fichier ou des fichiers donnés.

```
TIMEFORMAT='(TARGETFS) Compilation de file-5.11 en %R seconds ...'
time {
tar xzf ${CLFS}/download/file-5.37.tar.gz -C ${CLFS}/src &&
cd ${CLFS}/src/file-5.37 &&
./configure --prefix=/usr \
            --target=$TARGET \
            --host=$TARGET   \
            --disable-docs &&
make SHLIB_LIBS="-L${CLFS}/targetfs/usr/lib/ -lncursesw" &&
make DESTDIR=${CLFS}/targetfs SHLIB_LIBS="-L${CLFS}/targetfs/usr/lib/ -
lncursesw" install &&
cd "${CLFS}"
}
```

Installation de Readline

(TARGETFS) Compilation de readline-8.0 en 16.920 seconds ...^

Le paquet Readline est un ensemble de bibliothèques qui offrent des fonctionnalités d'édition de la ligne de commande et d'historique.

```
TIMEFORMAT='(TARGETFS) Compilation de readline-8.0 en %R seconds ...'
time {
tar xzf ${CLFS}/download/readline-8.0.tar.gz -C ${CLFS}/src &&
cd ${CLFS}/src/readline-8.0 &&
sed -i '/MV.*old/d' Makefile.in &&
sed -i '/{OLDSUFF}/c:' support/shlib-install &&
./configure --prefix=/usr      \
            --target="$TARGET" \
            --host="$TARGET"   \
            --disable-static &&
make SHLIB_LIBS="-L${CLFS}/targetfs/usr/lib -lncursesw" &&
make DESTDIR=${CLFS}/targetfs SHLIB_LIBS="-L${CLFS}/targetfs/usr/lib -
lncursesw" install &&
mv -v ${CLFS}/targetfs/usr/lib/lib{readline,history}.so.*
${CLFS}/targetfs/lib &&
chmod -v u+w ${CLFS}/targetfs/lib/lib{readline,history}.so.* &&
cd ${CLFS}/targetfs/usr/lib/ &&
ln -sfv ../../lib/$(readlink ${CLFS}/targetfs/usr/lib/libreadline.so)
libreadline.so &&
ln -sfv ../../lib/$(readlink ${CLFS}/targetfs/usr/lib/libhistory.so )
libhistory.so &&
cd "${CLFS}"
}
```

Installation de Bc

(TARGETFS) Compilation de bc-2.1.3 en 19.022 seconds ...

Le paquet Bc contient un langage de traitement des nombres à la précision de votre choix.

```
TIMEFORMAT='(TARGETFS) Compilation de bc-2.1.3 en %R seconds ...'
time {
tar xzf ${CLFS}/download/bc-2.1.3.tar.gz -C ${CLFS}/src &&
cd ${CLFS}/src/bc-2.1.3 &&
PREFIX=/usr CC=gcc CFLAGS="-std=c99" ./configure.sh -G -O3 &&
make &&
make DESTDIR=${CLFS}/targetfs install &&
cd "${CLFS}"
}
```

Installation de Bzip2

(TARGETFS) Compilation de bzip2-1.0.6 en 5.632 seconds ...

Le paquet Bzip2 contient des programmes de compression et décompression de fichiers. Compresser des fichiers texte avec bzip2 permet d'atteindre un taux de compression bien meilleur qu'avec l'outil gzip.

```
TIMEFORMAT='(TARGETFS) Compilation de bzip2-1.0.6 en %R seconds ...'
time {
tar xzf ${CLFS}/download/bzip2-1.0.6.tar.gz -C ${CLFS}/src &&
cd ${CLFS}/src/bzip2-1.0.6 &&
cp -v Makefile{,.orig} &&
sed -i 's@\(ln -s -f \)$(PREFIX)/bin/@\1@' Makefile &&
make -f Makefile-libbz2_so &&
make bzip2recover libbz2.a &&
cp bzip2-shared ${CLFS}/targetfs/bin/bzip2 &&
cp bzip2recover ${CLFS}/targetfs/bin &&
cp bzip2.1 ${CLFS}/targetfs/usr/share/man/man1 &&
cp bzlib.h ${CLFS}/targetfs/usr/include &&
cp -a libbz2.so* libbz2.a ${CLFS}/targetfs/lib &&
cd ${CLFS}/targetfs/bin &&
ln -svf bzip2 bunzip2 &&
ln -svf bzip2 bzcat &&
cd ${CLFS}/targetfs/usr/share/man/man1 &&
```

```
ln -svf bzip2.1 bunzip2.1 &&
ln -svf bzip2.1 bzcat.1 &&
ln -svf bzip2.1 bzip2recover.1 &&
cd "${CLFS}"
}
```

Installation de Sed

(TARGETFS) Compilation de sed-4.7 en 33.463 seconds ...

Le paquet Sed contient un éditeur de flux.

```
TIMEFORMAT='(TARGETFS) Compilation de sed-4.7 en %R seconds ...'
time {
tar xf ${CLFS}/download/sed-4.7.tar.xz -C ${CLFS}/src &&
cd ${CLFS}/src/sed-4.7 &&
sed -i 's/testsuite.panic-tests.sh//' Makefile.in &&
./configure --prefix=/usr --bindir=/bin --target="$TARGET" --host="$TARGET"
&&
make &&
make DESTDIR=${CLFS}/targetfs install &&
install -d -m755 ${CLFS}/targetfs/usr/share/doc/sed-4.7 &&
cd "${CLFS}"
}
```

Installation of Iana-Etc

(TARGETFS) Compilation de iana-etc-2.30 en 0.394 seconds ...

Le paquet Iana-Etc fournit des données pour les services et protocoles réseau.

```
TIMEFORMAT='(TARGETFS) Compilation de iana-etc-2.30 en %R seconds ...'
time {
tar xjf ${CLFS}/download/iana-etc-2.30.tar.bz2 -C ${CLFS}/src &&
cd ${CLFS}/src/iana-etc-2.30 &&
make STRIP=yes &&
make DESTDIR=${CLFS}/targetfs install &&
cd "${CLFS}"
}
```

Installation de Grep

(TARGETFS) Compilation de grep-3.3 en 42.644 seconds ...

Le paquet Grep contient des programmes de recherche à l'intérieur de fichiers.

```
TIMEFORMAT='(TARGETFS) Compilation de grep-3.3 en %R seconds ...'
time {
tar xf ${CLFS}/download/grep-3.3.tar.xz -C ${CLFS}/src &&
cd ${CLFS}/src/grep-3.3 &&
./configure --prefix=/usr --bindir=/bin --target="$TARGET" --host="$TARGET"
&&
make &&
make DESTDIR=${CLFS}/targetfs install &&
cd "${CLFS}"
}
```

Installation de Gperf

(TARGETFS) Compilation de gperf-3.1 en 8.351 seconds ...

Gperf génère une fonction de hachage parfait à partir d'un trousseau.

```
TIMEFORMAT='(TARGETFS) Compilation de gperf-3.1 en %R seconds ...'
time {
tar xzf ${CLFS}/download/gperf-3.1.tar.gz -C ${CLFS}/src &&
cd ${CLFS}/src/gperf-3.1 &&
./configure --prefix=/usr --target="$TARGET" --host="$TARGET" --disable-docs
&&
make &&
make DESTDIR=${CLFS}/targetfs install &&
cd "${CLFS}"
}
```

Installation de Automake

(TARGETFS) Compilation de automake-1.16.1 en 5.046 seconds ...

Le paquet Automake contient des programmes de génération de Makefile à utiliser avec Autoconf.

```
TIMEFORMAT='(TARGETFS) Compilation de automake-1.16.1 en %R seconds ...'
time {
tar xf ${CLFS}/download/automake-1.16.1.tar.xz -C ${CLFS}/src &&
cd ${CLFS}/src/automake-1.16.1 &&
./configure --prefix=/usr --target="$TARGET" --host="$TARGET" --disable-docs
&&
make &&
make DESTDIR=${CLFS}/targetfs install &&
cd "${CLFS}"
}
```

Installation de Gawk

(TARGETFS) Compilation de gawk-4.2.1 en 59.182 seconds ...

Le paquet Gawk contient des programmes de manipulation de fichiers texte.

```
TIMEFORMAT='(TARGETFS) Compilation de gawk-4.2.1 en %R seconds ...'
time {
tar xf ${CLFS}/download/gawk-4.2.1.tar.xz -C ${CLFS}/src &&
cd ${CLFS}/src/gawk-4.2.1 &&
sed -i 's/extras//' Makefile.in &&
./configure --prefix=/usr --target="$TARGET" --host="$TARGET" &&
make &&
make DESTDIR=${CLFS}/targetfs install &&
cd "${CLFS}"
}
```

Installation de Make

(TARGETFS) Compilation de make-4.2.1 en 20.660 seconds ...

Le paquet Make contient un programme pour compiler des paquets.

```
TIMEFORMAT='(TARGETFS) Compilation de make-4.2.1 en %R seconds ...'
time {
tar xjf ${CLFS}/download/make-4.2.1.tar.bz2 -C ${CLFS}/src &&
cd ${CLFS}/src/make-4.2.1 &&
sed -i '211,217 d; 219,229 d; 232 d' glob/glob.c &&
./configure --prefix=/usr --without-guile --target="$TARGET" --
host="$TARGET" &&
make &&
```

```
make DESTDIR=${CLFS}/targetfs install &&
cd "${CLFS}"
}
```

Installation de Patch

(TARGETFS) Compilation de patch-2.7.6 en 43.358 seconds ...

Le paquet Patch contient un programme permettant de modifier et de créer des fichiers en appliquant un fichier correctif (appelé habituellement « patch ») créé généralement par le programme diff.

```
TIMEFORMAT='(TARGETFS) Compilation de patch-2.7.6 en %R seconds ...'
time {
tar xf ${CLFS}/download/patch-2.7.6.tar.xz -C ${CLFS}/src &&
cd ${CLFS}/src/patch-2.7.6 &&
./configure --prefix=/usr --target="$TARGET" --host="$TARGET" &&
make &&
make DESTDIR=${CLFS}/targetfs install &&
cd "${CLFS}"
}
```

Installation de BusyBox

(TARGETFS) Compilation de busybox-1.31.0 en 120.333 seconds ...

BusyBox regroupe des versions minuscules de nombreux utilitaires UNIX courants en un seul petit exécutable, qui remplace la plupart des utilitaires que l'on trouve habituellement dans GNU fileutils, shellutils, etc. Cependant, les options incluses fournissent les fonctionnalités attendues et se comportent de manière très similaire à leurs homologues GNU: BusyBox fournit un environnement assez complet pour tout système de petite taille ou embarqué.

Attention: Le Makefile de Busybox utilise la même variable d'environnement `KBUILD_OUTPUT` que celle utilisée pour extraire les headers du noyau. Avant de le compiler il faut donc s'assurer que cette variable est écrasée par `unset KBUILD_OUTPUT`

```
TIMEFORMAT='(TARGETFS) Compilation de busybox-1.31.0 en %R seconds ...'
time {
unset KBUILD_OUTPUT &&
tar xjf ${CLFS}/download/busybox-1.31.0.tar.bz2 -C ${CLFS}/src &&
cd ${CLFS}/src/busybox-1.31.0 &&
make distclean &&
make ARCH="${ARCH}" CROSS_COMPILE="${TARGET}-" defconfig &&
cp .config .config.orig &&
```

```
sed -e "s:# USING_CROSS_COMPILER.*:USING_CROSS_COMPILER=y:" \
    -e "/^CROSS_COMPILER_PREFIX/s:=.*:=\"${TARGET}-\":" \
    .config.orig > .config &&
sed -i 's/\(CONFIG_\)\(.*\)\(INETD\)\(.*\)=y/# \1\2\3\4 is not set/g'
.config &&
sed -i 's/\(CONFIG_IFPLUGD\)=y/# \1 is not set/' .config &&
sed -i 's/\(CONFIG_FEATURE_WTMP\)=y/# \1 is not set/' .config &&
sed -i 's/\(CONFIG_FEATURE_UTMP\)=y/# \1 is not set/' .config &&
sed -i 's/\(CONFIG_UDPSVD\)=y/# \1 is not set/' .config &&
sed -i 's/\(CONFIG_TCPSVD\)=y/# \1 is not set/' .config &&
make CROSS="${TARGET}-" PREFIX="${CLFS/targetfs}"  EXTRA_CFLAGS="${BUILD}"
LDFLAGS=-static busybox &&
make PREFIX="${CLFS}/targetfs" install &&
cd "${CLFS}"
}
```

Installation de Dropbear

(TARGETFS) Compilation de dropbear-2019.78 en 49.963 seconds ..

Dropbear est un serveur et un client SSH 2 relativement petit. Dropbear a une faible empreinte mémoire qui convient aux environnements à mémoire limitée, tout en conservant les mêmes fonctionnalités qu'OpenSSH. Il ne dépend pas d'OpenSSL et dispose d'une licence de type MIT.

```

TIMEFORMAT='(TARGETFS) Compilation de dropbear-2019.78 en %R seconds ...'
time {
tar xjf ${CLFS}/download/dropbear-2019.78.tar.bz2 -C ${CLFS}/src &&
cd ${CLFS}/src/dropbear-2019.78 &&
sed -i 's/.*mandir.*//g' Makefile.in &&
cp -av ${CLFS}/targetfs/include/{zlib.h,zconf.h}
${CLFS}/src/dropbear-2019.78 &&
CC="${CC} -Os" ./configure --with-sysroot="$CLFS/targetfs" --prefix=/usr --
host=${TARGET} --target=${TARGET} --with-zlib=${CLFS}/targetfs/lib --enable-
static &&
make MULTI=1 \
  PROGRAMS="dropbear dbclient dropbearkey dropbearconvert scp" &&
make MULTI=1 \
  PROGRAMS="dropbear dbclient dropbearkey dropbearconvert scp" \
  install DESTDIR=${CLFS}/targetfs &&
cd $CLFS/targetfs/usr/sbin/ &&
ln -svf ../bin/dropbearmulti dropbear &&
ln -svf ../bin/dropbearmulti dbclient &&
$CLFS/targetfs/usr/bin/ &&
ln -svf dropbearmulti dropbearkey
ln -svf dropbearmulti dropbearconvert
ln -svf dropbearmulti scp
install -dv ${CLFS}/targetfs/etc/dropbear
```

```
cd "${CLFS}"
}
```

Nettoyage des symboles de déboggage

Lors de la compilation d'un programme, GCC ajoute des symboles au fichier binaire pour aider le développeur lors du débogage.Contrairement aux idées reçues, GCC écrit des symboles dans un fichier objet même en mode release (avec le commutateur -O3).

La commande `nm` permet d'afficher les symboles de débooggage des binaires.

```
nm ${CLFS}/targetfs/usr/bin/*

...
08064330 T write_ieee_debugging_info
080a93a0 t write_obj_attribute.part.2
080686d0 T write_stabs_in_sections_debugging_info
080aae30 t write_value.isra.4
080f01b0 T writeargv
08078b30 t writesym
08078830 t writevalue
080f5fa0 T xcalloc
080f5ea0 T xexit
080f5f70 T xmalloc
080f5f00 T xmalloc_failed
080f5ec0 T xmalloc_set_program_name
080f5ff0 T xrealloc
080f6040 T xstrdup
080f6070 T xstrerror
081602a0 b xstrerror_buf
080f60b0 T xstrndup
080e0070 T zError
0811d940 R z_errmsg
080e0090 T zcalloc
080e00b0 T zcfree
080e0060 T zlibCompileFlags
080e0050 T zlibVersion
```

Les symboles peuvent être supprimés du binaire à l'aide de la commande strip, le binaire qui en résulte est plus petit, ce qui signifie qu'il utilise moins de mémoire et qu'il est donc probablement plus rapide.

Par exemple, avant le nettoyage des symboles de déboggages le système de de fichier construit devrait contenir quelque chose comme 697M, donc quand on va charger l'initram en mémoire cela va prendre 697Mo.

```
501M    ./usr/libexec/gcc/i686-linux-musl/8.2.0
697M    .
```

Si on applique strip sur le contenu de `/usr/libexec/gcc/i686-linux-musl/8.2.0` qui représente à lui seul 501Mo ...

```
for binary in ${CLFS}/targetfs/usr/libexec/gcc/i686-linux-musl/8.2.0/*
 do
   strip -v ${binary}
done
```

... on reqmarque que le contenu de `/usr/libexec/gcc/i686-linux-musl/8.2.0` ne représent plus que 81Mo de mémoire.

```
81M ./usr/libexec/gcc/i686-linux-musl/8.2.0
277M
```

L'application de cette stratégie à l'ensemble du système permet de mieux en optimimiser la réactivité.

On peut vérifier l'absence des symboles en utilisant de nouveau `nm ./bin/*`.

Configuration de mdev

Mdev est un remplacement de udev dans BusyBox, avec une base de règles différente.

```
cat > ${CLFS}/targetfs/etc/mdev.conf<< "EOF"
# /etc/mdev/conf

# Devices:
# Syntax: %s %d:%d %s
# devices user:group mode

81M ./usr/libexec/gcc/i686-linux-musl/8.2.0
277M

# Null existe déjà, il faut donc changer de propriétaire avec commande
null    root:root 0666  @chmod 666 $MDEV
zero    root:root 0666
grsec   root:root 0660
full    root:root 0666

random  root:root 0666
urandom root:root 0444
hwrandom root:root 0660

# La console existe déjà, il faut donc changer de propriétaire avec la
commande
#console        root:tty 0600   @chmod 600 $MDEV && mkdir -p vc && ln -sf
../$MDEV vc/0
```

```
console root:tty 0600 @mkdir -pm 755 fd && cd fd && for x in 0 1 2 3 ; do ln
-sf /proc/self/fd/$x $x; done

fd0     root:floppy 0660
kmem    root:root 0640
mem     root:root 0640
port    root:root 0640
ptmx    root:tty 0666

# ram.*
ram([0-9]*)     root:disk 0660 >rd/%1
loop([0-9]+)    root:disk 0660 >loop/%1
sd[a-z].*       root:disk 0660 */lib/mdev/usbdisk_link
hd[a-z][0-9]*   root:disk 0660 */lib/mdev/ide_links
md[0-9]         root:disk 0660

tty             root:tty 0666
tty[0-9]        root:root 0600
tty[0-9][0-9]   root:tty 0660
ttyS[0-9]*      root:tty 0660
pty.*           root:tty 0660
vcs[0-9]*       root:tty 0660
vcsa[0-9]*      root:tty 0660

ttyLTM[0-9]     root:dialout 0660 @ln -sf $MDEV modem
ttySHSF[0-9]    root:dialout 0660 @ln -sf $MDEV modem
slamr           root:dialout 0660 @ln -sf $MDEV slamr0
slusb           root:dialout 0660 @ln -sf $MDEV slusb0
fuse            root:root  0666

# dri device
card[0-9]       root:video 0660 =dri/

# alsa sound devices and audio stuff
pcm.*           root:audio 0660 =snd/
control.*       root:audio 0660 =snd/
midi.*          root:audio 0660 =snd/
seq             root:audio 0660 =snd/
timer           root:audio 0660 =snd/

adsp            root:audio 0660 >sound/
audio           root:audio 0660 >sound/
dsp             root:audio 0660 >sound/
mixer           root:audio 0660 >sound/
sequencer.*     root:audio 0660 >sound/

# misc stuff
agpgart         root:root 0660  >misc/
psaux           root:root 0660  >misc/
rtc             root:root 0664  >misc/
```

```
# input stuff
event[0-9]+     root:root 0640 =input/
mice            root:root 0640 =input/
mouse[0-9]      root:root 0640 =input/
ts[0-9]         root:root 0600 =input/

# v4l stuff
vbi[0-9]        root:video 0660 >v4l/
video[0-9]      root:video 0660 >v4l/

# dvb stuff
dvb.*           root:video 0660 */lib/mdev/dvbdev

# load drivers for usb devices
usbdev[0-9].[0-9]       root:root 0660 */lib/mdev/usbdev
usbdev[0-9].[0-9]_.*    root:root 0660

# net devices
tun[0-9]*       root:root 0600 =net/
tap[0-9]*       root:root 0600 =net/

# zaptel devices
zap(.*)         root:dialout 0660 =zap/%1
dahdi!(.*)      root:dialout 0660 =dahdi/%1

# raid controllers
cciss!(.*)      root:disk 0660 =cciss/%1
ida!(.*)        root:disk 0660 =ida/%1
rd!(.*)         root:disk 0660 =rd/%1

sr[0-9]         root:cdrom 0660 @ln -sf $MDEV cdrom

# hpilo
hpilo!(.*)      root:root 0660 =hpilo/%1

# xen stuff
xvd[a-z]        root:root 0660 */lib/mdev/xvd_links
EOF
```

Création de /etc/profile

```
cat > ${CLFS}/targetfs/etc/profile << "EOF"
# /etc/profile

# Set the initial path
export PATH=/bin:/usr/bin

if [ `id -u` -eq 0 ] ; then
        PATH=/bin:/sbin:/usr/bin:/usr/sbin
```

```
        unset HISTFILE
fi

# Setup some environment variables.
export USER=`id -un`
export LOGNAME=$USER
export HOSTNAME=`/bin/hostname`
export HISTSIZE=1000
export HISTFILESIZE=1000
export PAGER='/bin/more '
export EDITOR='/bin/vi'

# End /etc/profile
EOF
```

Création de /etc/inittab

```
cat > ${CLFS}/targetfs/etc/inittab<< "EOF"
# /etc/inittab

::sysinit:/etc/rc.d/startup

tty1::respawn:/sbin/getty 38400 tty1
tty2::respawn:/sbin/getty 38400 tty2
tty3::respawn:/sbin/getty 38400 tty3
tty4::respawn:/sbin/getty 38400 tty4
tty5::respawn:/sbin/getty 38400 tty5
tty6::respawn:/sbin/getty 38400 tty6

# Met un getty sur la ligne série (pour un terminal). Décommenter cette
ligne pour
# utiliser une console série sur ttyS0, ou décommenter et régler pour
utiliser une
# console série sur un autre port série.
#::respawn:/sbin/getty -L ttyS0 115200 vt100

::shutdown:/etc/rc.d/shutdown
::ctrlaltdel:/sbin/reboot
EOF
```

Définition du Hostname

```
echo "[clfs]" > ${CLFS}/targetfs/etc/HOSTNAME
```

Création du fichier /etc/hosts

```

```

```
cat > ${CLFS}/targetfs/etc/hosts << "EOF"
# Begin /etc/hosts (network card version)

127.0.0.1 localhost
[192.168.1.1] [<HOSTNAME>.example.org] [HOSTNAME]

# End /etc/hosts (network card version)
EOF
```

Ou si aucune carte réseau ne va être configurée:

```
cat > ${CLFS}/targetfs/etc/hosts << "EOF"
# Begin /etc/hosts (no network card version)

127.0.0.1 [<HOSTNAME>.example.org] [HOSTNAME] localhost

# End /etc/hosts (no network card version)
EOF
```

Création de fichiers de configuration d'interface réseau

```
mkdir -pv ${CLFS}/targetfs/etc/network/if-{post-{up,down},pre-
{up,down},up,down}.d
mkdir -pv ${CLFS}/targetfs/usr/share/udhcpc
```

```
cat > ${CLFS}/targetfs/etc/network/interfaces << "EOF"
auto eth0
iface eth0 inet dhcp
EOF
```

```
cat > ${CLFS}/targetfs/usr/share/udhcpc/default.script << "EOF"
#!/bin/sh
# udhcpc Interface Configuration
# Based on http://lists.debian.org/debian-boot/2002/11/msg00500.html
# udhcpc script edited by Tim Riker <Tim@Rikers.org>

[ -z "$1" ] && echo "Error: should be called from udhcpc" && exit 1

RESOLV_CONF="/etc/resolv.conf"
[ -n "$broadcast" ] && BROADCAST="broadcast $broadcast"
[ -n "$subnet" ] && NETMASK="netmask $subnet"

case "$1" in
        deconfig)
                /sbin/ifconfig $interface 0.0.0.0
                ;;
```

```
        renew|bound)
                /sbin/ifconfig $interface $ip $BROADCAST $NETMASK

                if [ -n "$router" ] ; then
                        while route del default gw 0.0.0.0 dev $interface ;
do
                                true
                        done

                        for i in $router ; do
                                route add default gw $i dev $interface
                        done
                fi

                echo -n > $RESOLV_CONF
                [ -n "$domain" ] && echo search $domain >> $RESOLV_CONF
                for i in $dns ; do
                        echo nameserver $i >> $RESOLV_CONF
                done
                ;;
esac

exit 0
EOF

chmod +x ${CLFS}/targetfs/usr/share/udhcpc/default.script
```

Changer la propriété du système CLFS

```
chown -Rv root:root ${CLFS}/targetfs
chgrp -v 13 ${CLFS}/targetfs/var/log/lastlog
```

Confection de l'image rootfs

```
install -dv ${CLFS}/build
cd ${CLFS}/targetfs
tar jcfv ${CLFS}/build/clfs-embedded.tar.bz2 *
```

Confection du kernel

Configuration du Kernel

```
cd ${CLFS}/src/linux-4.20.12
make mrproper
make ARCH=${CLFS_ARCH} CROSS_COMPILE=${CLFS_TARGET}- menuconfig
```

Construction du kernel

```
make ARCH=${CLFS_ARCH} CROSS_COMPILE=${CLFS_TARGET}-
```

Pour intégrer des modules:

```
make ARCH=${CLFS_ARCH} CROSS_COMPILE=${CLFS_TARGET}- \
    INSTALL_MOD_PATH=${CLFS}/targetfs modules_install
```

Packages optionnels

BusyBox regroupe des versions minuscules de nombreux utilitaires UNIX dans un seul et même petit exécutable, qui fournit des remplacements minimalistes pour la plupart des utilitaires que l'on trouve habituellement dans GNU coreutils, util-linux, etc. Cependant, pour les les options non incluses il faut installer leurs homologues GNU.

NFS

Installation de NFS Utilities

Fix Me! Le paquet NFS Utilities contient le serveur en espace utilisateur et le client nécessaires pour utiliser les possibilités NFS du noyau. NFS est un protocole qui permet le partage de systèmes de fichiers sur un réseau.

Les paquets libtirpc et rpcsvc-proto sont requis.

Installation de libtirpc

Le paquet libtirpc contient des bibliothèques qui supportent des programmes utilisant l'API de Remote Procedure Call (RPC). Il remplace le RPC, mais pas les entrées de la bibliothèque NIS qui se trouvaient dans glibc.

```
TIMEFORMAT='(TARGETFS) Compilation de libtirpc-1.1.4 en %R seconds ...'
time {
tar xyf ${CLFS}/download/libtirpc-1.1.4.tar.bz2 -C ${CLFS}/src &&
cd ${CLFS}/src/libtirpc-1.1.4 &&
```

```
./configure --prefix=/usr     \
            --target=$TARGET  \
            --host=$TARGET    \
            --sysconfdir=/etc \
            --disable-static  \
            --disable-gssapi &&
make &&
make DESTDIR=${CLFS}/targetfs install &&
mv -v /usr/lib/libtirpc.so.* /lib &&
cd /usr/lib/ &&
ln -sfv libtirpc.so.3.0.0 libtirpc.so &&
cd "${CLFS}"
}
```

Installation de rpcsvc-proto

Le paquet rpcsvc-proto contient les fichiers et les en-têtes rcpsvc protocol.x, précédemment inclus dans glibc et qui ne sont pas inclus dans le paquet libtirpc de remplacement, ainsi que le programme rpcgen.

```
TIMEFORMAT='(TARGETFS) Compilation de rpcsvc-proto-1.4 en %R seconds ...'
time {
tar xyvf rpcsvc-proto-1.4.tar.gz -C ${CLFS}/src &&
cd ${CLFS}/src/rpcsvc-proto-1.4 &&
./configure --sysconfdir=/etc --target=$TARGET --host=$TARGET &&
make &&
make DESTDIR=${CLFS}/targetfs install &&
cd "${CLFS}"
}
```

Installation de nfs-utils

```
TIMEFORMAT='(TARGETFS) Compilation de nfs-utils-2.3.3 en %R seconds ...'
time {
tar xyf ${CLFS}/download/nfs-utils-2.3.3.tar.xz -C ${CLFS}/src &&
cd ${CLFS}/src/nfs-utils-2.3.3
./configure --prefix=/usr     \
            --target=$TARGET  \
            --host=$TARGET    \
            --sysconfdir=/etc \
            --sbindir=/sbin   \
            --disable-nfsv4   \
            --disable-gss &&
make &&
make DESTDIR=${CLFS}/targetfs install &&
mv -v ${CLFS}/targetfs/sbin/start-statd ${CLFS}/targetfs/usr/sbin  &&
chmod u+w,go+r ${CLFS}/targetfs/sbin/mount.nfs &&
```

```
chown nobody.nogroup ${CLFS}/targetfs/var/lib/nfs &&
cd "${CLFS}"
}
```

Configuration du kernel

Dans le kernel activer les options suivantes

```
File systems  --->
  [*] Network File Systems  --->         [CONFIG_NETWORK_FILESYSTEMS]
    <*/M> NFS client support             [CONFIG_NFS_FS]
```

Programmation

Installation de Bison

Bison, est un générateur d'analyseur syntaxique, il lit les spécifications d'un langage, signale les ambiguïtés d'analyse, et génère un analyseur syntaxique (en C, C++ ou Java) qui lit chaque séquence de chaînes de caractères et décide si cette séquence est conforme à la syntaxe spécifiée par la grammaire.

```
TIMEFORMAT='(TARGETFS) Compilation de bison-3.3.2 en %R seconds ...'
time {
tar xf ${CLFS}/download/bison-3.3.2.tar.xz -C ${CLFS}/src &&
cd ${CLFS}/src/bison-3.3.2 &&
./configure --prefix=/usr --target=$TARGET --host=$TARGET &&
make &&
make DESTDIR=${CLFS}/targetfs install &&
cd "${CLFS}"
}
```

Installation de Flex

Le paquet Flex contient un outil de génération de programmes reconnaissant des modèles de texte.

```
TIMEFORMAT='(TARGETFS) Compilation de flex-2.6.4 en %R seconds ...'
time {
tar xzf ${CLFS}/download/flex-2.6.4.tar.gz -C ${CLFS}/src &&
cd ${CLFS}/src/flex-2.6.4.tar.gz &&
HELP2MAN=/tools/bin/true \
```

```
./configure --prefix=/usr --target=$TARGET --host=$TARGET &&
make &&
make DESTDIR=${CLFS}/targetfs install &&
cd ${CLFS}/targetfs/usr/bin/ &&
ln -s flex lex &&
cd "${CLFS}"
}
```

Utilitaires

Installation de Bash

Le paquet Bash contient le shell Bourne-Again.

```
TIMEFORMAT='(TARGETFS) Compilation de bash-5.0 en %R seconds ...'
time {
tar xzf ${CLFS}/download/bash-5.0.tar.gz -C ${CLFS}/src &&
cd ${CLFS}/src/bash-5.0 &&
./configure --prefix=/usr             \
            --target=$TARGET          \
            --host=$TARGET            \
            --without-bash-malloc     \
            --with-installed-readline &&
make &&
chown -Rv nobody . &&
make DESTDIR=${CLFS}/targetfs install &&
mv -vf ${CLFS}/targetfs/usr/bin/bash ${CLFS}/targetfs/bin &&
cd "${CLFS}"
}
```

Installation de Vim

Le paquet Vim contient un puissant éditeur de texte.

```
TIMEFORMAT='(TARGETFS) Compilation de vim-8.1 en %R seconds ...'
time {
tar ${CLFS}/download/vim-8.1.tar.bz2  -C ${CLFS}/src &&
cd ${CLFS}/src/vim-8.1 &&
echo '#define SYS_VIMRC_FILE "/etc/vimrc"' >> src/feature.h  &&
./configure --prefix=/usr --target=$TARGET --host=$TARGET &&
make &&
make DESTDIR=${CLFS}/targetfs install &&
cd ${CLFS}/targetfs/usr/bin/ &&
```

```
ln -sv vim vi &&
cd ${CLFS}/targetfs/usr/ &&
for L in  share/man/{,*/}man1/vim.1; do
    ln -sv vim.1 $(dirname $L)/vi.1
done &&
rm ${CLFS}/targetfs/usr/share/vim/vim81/doc &&
cd "${CLFS}"
}
```

```
cat > /etc/vimrc << "EOF"
" Begin /etc/vimrc

" Assure que les paramètres par défaut sont définis avant de personnaliser
les paramètres, et non après.
source $VIMRUNTIME/defaults.vim
let skip_defaults_vim=1

set nocompatible
set backspace=2
set mouse=
syntax on
if (&term == "xterm") || (&term == "putty")
  set background=dark
endif

" End /etc/vimrc
EOF
```

```
cp ${CLFS}/src/fr.utf-8.spl ${CLFS}/targetfs/usr/share/vim/vim81/spell/
cp ${CLFS}/src/spell/fr.utf-8.sug
${CLFS}/targetfs/usr/share/vim/vim81/spell/
set spelllang=fr,en
set spell
```