```
#!/bin/bash

VERT="\033[1;32m"
NORMAL="\033[0m"
ROUGE="\033[1;31m"
CYAN="\\033[1;36m"
BINUTILS="binutils-2.22"
KERNEL="linux-3.5.3"
GCC="gcc-4.7.2"

reponse=0
clear
echo ""
echo ""
echo "*************************************************************************"
echo -e "Bienvenue à toi $VERT $LOGNAME $NORMAL dans l'outil de génération d'une cross-tool chain"
echo "*************************************************************************"
while [ $reponse -ne 11 ]
        do

echo ""
echo -e "       ____________________________________________________________________"
echo -e""
echo -e "        | $CYAN 1-$NORMAL       RTFM $VERT(Obligatoire)$NORMAL                                |"
echo -e "        | $CYAN 2-$NORMAL       Construction de l'arborescence                |"
echo -e "        | $CYAN 3-$NORMAL       Téléchargement des archives                   |"
echo -e "        | $CYAN 4-$NORMAL       Décompresser les Archives                     |"
echo -e "        | $CYAN 5-$NORMAL       Conpilation des BINUTILS                      |"
echo -e "        | $CYAN 6-$NORMAL       Conpilation Kernel HEADERS                    |"
echo -e "        | $CYAN 7-$NORMAL       Conpilation GCC-bootstrap                     |"
echo -e "        | $CYAN 8-$NORMAL       Conpilation En-tete eGlibC                    |"
echo -e "        | $CYAN 9-$NORMAL       Compilation eGlibC                            |"
echo -e "        | $CYAN 10-$NORMAL      Compilation GCC                               |"
echo -e "        | $CYAN 11-$NORMAL      Hello WORLD                                   |"
echo -e "       ____________________________________________________________________ "
echo -e "\n"
echo -ne "$CYAN Etape n° : $NORMAL "
read reponse
echo ""
case $reponse in

1)
clear
echo -e "    $CYAN         ------------ Sous Menu Préambule -------------    $NORMAL  "
echo ""
echo " ----> Pré-requis pour le bon fonctionnement de la génération de la cross-tool "
echo ""
echo -e " ----> Rendez-vous sur la page $CYAN https://github.com/texierp/TP-LINUX$NORMAL"
echo ""
echo  -e " $ROUGE----> Pour installer un paquet manuellement : su -c 'aptitude install nom du paquet'
$NORMAL"

rep=0
while [ $rep -ne 2 ]
        do

echo ""
echo -e "
_____________________________________________________________________________________________"
echo -e""
echo -e "        | $CYAN 1-$NORMAL       Je suis débutant je préfére générer l'installation des
paquets :)|"
echo -e "        | $CYAN 2-$NORMAL       Je suis expert LINUX je vais le faire moi meme :) je tape ctrl
+c |"
echo -e "
_____________________________________________________________________________________________"
echo -e "\n"
echo -e "\n"
echo -ne "$CYAN Entrez votre Choix ---> n° : $NORMAL "
read rep
echo ""
case $rep in
```

```
1)
#-------------------------------------------#
#         Installation des Paquets          #
#-------------------------------------------#

su -c 'aptitude -y install gawk libppl-dev bison libmpc-dev make build-essential libsvn-dev flex
libmpfr-dev lib32mpfr4 libgmp-dev ppl m4 autogen subversion texinfo diffutils autoconf makeinfo cloog-
ppl libcloog-ppl-dev pkg-config dconf-tools libpthread-stub0-dev libevent-pthreads-2.0-5 pthread
libqt4-dev gperf libpthread-workqueue-dev'
;;
2)
        exit 0;;
esac
done

;;

2)
echo ""
echo ""
echo "|-------------------------------------------------------|"
echo "|   HOME  ---->  eGcross  ----->    sources/archives    |"
echo "|                    |--------->    build               |"
echo "|                    |--------->    arm/sysroot         |"
echo "|-------------------------------------------------------|"

echo ""
echo ""
#-------------------------------------------#
#         On crée donc l'arborescence       #
#-------------------------------------------#

if [ -d eGcross ]    # On test
then
        rm -r eGcross
fi

if [ ! -d eGcross ]  # On test
then
        mkdir eGcross
        cd eGcross
        mkdir -p sources/archives
        mkdir build
        mkdir -p arm/sysroot
fi

echo "L'arborescence vient d'etre crée !!!! On peut commencer à travailler "
echo $LOGNAME


#-------------------------------------------#
#  On définie les variables d'environnement #
#-------------------------------------------#

cd $HOME
echo "############ export des variables d'environnement ########## "
export THREADS=$(egrep -c 'processor' /proc/cpuinfo)           # Nombres de THREADS
export DUMP=$(gcc -dumpmachine)                                # Nom de la machine
export SRCDIR=$HOME/eGcross/sources                            # Dossier sources
export BUILDDIR=$HOME/eGcross/build                            # Dossier paquets compilés
export TARGET=arm-none-linux-gnueabi                           # ARCH Target
export BUILD=$DUMP                                             # Arch build
export INSTALLDIR=$HOME/eGcross/arm                            # Dossier contenant la Cross
export SYSROOTDIR=$HOME/eGcross/arm/sysroot                    # Dossier lib et header
echo ""
echo ""
;;

3)
echo ""
echo -e " $ROUGE-Aide mémoire$NORMAL: http://www.kernel.org ,http://www.eglibc.org/ et http://
```

```
ftp.gnu.org"
echo ""
echo -e " $VERT-eGlibc"
echo -e " -GCC"
echo -e " -Le KERNEL"
echo -e " -Binutils$NORMAL"

#-------------------------------------------------#
#  Télechargement des archives et source (eGlibC)  #
#-------------------------------------------------#

echo ""
cd $SRCDIR
svn co http://www.eglibc.org/svn/branches/eglibc-2_16 eglibc-2.16    # On récupére la branche eGlibC
2.16
cd eglibc-2.16                                                       # On copie le fichier ports/ dans
libc/
cp -R ports/ libc/
echo -e "  $VERT Exit retourné $NORMAL "$?
cd ..
cd archives
wget http://ftp.gnu.org/gnu/binutils/$BINUTILS.tar.bz2
echo -e " $VERT-Téléchargement BINUTILS terminé$NORMAL"
echo ""
wget http://www.kernel.org/pub/linux/kernel/v3.x/$KERNEL.tar.bz2
echo -e " $VERT-Téléchargement KERNEL terminé$NORMAL"
echo ""
wget http://ftp.gnu.org/gnu/gcc/$GCC/$GCC.tar.bz2
echo -e " $VERT-Téléchargement GCC terminé$NORMAL"
echo ""
;;

4)
#--------------------------#
#      On décompresse      #
#--------------------------#

#-------------------------------------------------------------
tar xvjf $BINUTILS.tar.bz2
echo -e " $VERT-Extraction BINUTILS réussi$NORMAL"
mv $BINUTILS ../
echo -e " $VERT-Déplacement du dossier réussi$NORMAL"
#-------------------------------------------------------------
tar xvjf $KERNEL.tar.bz2
echo -e " $VERT-Extraction KERNEL réussi$NORMAL"
mv $KERNEL ../
echo -e " $VERT-Déplacement du dossier réussi$NORMAL"
#-------------------------------------------------------------
tar xvjf $GCC.tar.bz2
echo -e " $VERT-Extraction GCC réussi$NORMAL"
mv $GCC ../
echo -e " $VERT-Déplacement du dossier réussi$NORMAL"
#-------------------------------------------------------------

echo ""
echo "Il faut détruire le dossier archives"
cd ..
rm -r archives
if [ ! -d archives ]    # On test
then
        echo "Destruction du dossier repository 'archives' réussi"
fi
;;

5)
#--------------------------#
#  Compilation des sources  #
#--------------------------#
# BINUTILS
cd ..
cd build
mkdir $BUILDDIR/binutils
```

```
cd $BUILDDIR/binutils
echo -e " $VERT-Compilation BINUTILS$NORMAL"
../../sources/$BINUTILS/configure \
        --disable-werror \
        --build=$BUILD \
        --target=$TARGET \
        --with-sysroot=$SYSROOTDIR \
        --prefix=$INSTALLDIR

echo -e "  $VERT Exit retourné $NORMAL ---> "$?
make -j $THREADS
echo -e "  $VERT Exit retourné $NORMAL ---> "$?
make install
echo -e "  $VERT Exit retourné $NORMAL ---> "$?
echo -e " $VERT-Compilation BINUTILS réussi $NORMAL"
;;


6)
# KERNEL
cd $SRCDIR/$KERNEL
make mrproper
make ARCH=arm integrator_defconfig
echo -e "  $VERT Exit retourné $NORMAL ---> "$?
make ARCH=arm headers_check
echo -e "  $VERT Exit retourné $NORMAL ---> "$?
make ARCH=arm INSTALL_HDR_PATH=$INSTALLDIR/sysroot/usr headers_install
echo -e "  $VERT Exit retourné $NORMAL ---> "$?
echo -e " $VERT-Compilation KERNEL réussi $NORMAL"
;;


7)
# GCC-minimaliste
mkdir $BUILDDIR/gcc-bootstrap
cd $BUILDDIR/gcc-bootstrap

../../sources/gcc-4.7.2/configure \
        --host=$BUILD \
        --build=$BUILD \
        --target=$TARGET \
        --prefix=$INSTALLDIR \
        --without-headers \
        --enable-bootstrap \
        --enable-languages=c \
        --disable-threads \
        --enable-__cxa-atexit \
        --disable-libmudflap \
        --with-gnu-as \
        --with-gnu-ld \
        --with-newlib \
        --disable-libssp \
        --disable-libgomp \
        --disable-nls \
        --disable-shared

echo -e "  $VERT Exit retourné $NORMAL ---> "$?
make all-gcc install-gcc                        # Cette commande précédente crée le compilateur de
base et l'installe dans le répertoire  $INSTALLDIR
echo -e "  $VERT Exit retourné $NORMAL ---> "$?
make all-target-libgcc install-target-libgcc    # Nous lançons à présent la construction d'une
bibliothèque de base utilisée par GCC pour produire du code
echo -e "  $VERT Exit retourné $NORMAL ---> "$?
echo -e " $VERT-Compilation GCC-bootstrap réussi $NORMAL"
ln -s $INSTALLDIR/lib/gcc/arm-none-linux-gnueabi/4.7.2/libgcc.a $INSTALLDIR/lib/gcc/arm-none-linux-
gnueabi/4.7.2/libgcc_sh.a
;;


8)
# en-tete eglibc
export CROSS=arm-none-linux-gnueabi
```

```
export CC=${CROSS}-gcc
export LD=${CROSS}-ld
export AS=${CROSS}-as
export AR=${CROSS}-ar
export PATH=$INSTALLDIR/bin:$PATH
export RANLIB=${CROSS}-ranlib

mkdir $BUILDDIR/libc-header
cd $BUILDDIR/libc-header
echo "libc_cv_forced_unwind=yes" > config.cache
echo "libc_cv_c_cleanup=yes" >> config.cache
../../sources/eglibc-2.16/libc/configure \
        --build=$BUILD \
        --host=$TARGET \
        --prefix=/usr \
        --with-headers=$SYSROOTDIR/usr/include \
        --config-cache \
        --enable-kernel=3.5.3 \
        --disable-profile --without-gd --without-cvs --enable-add-ons=ports,nptl

echo -e "  $VERT Exit retourné $NORMAL ---> "$?
make -k install-headers cross_compiling=yes install_root=$SYSROOTDIR
echo -e "  $VERT Exit retourné $NORMAL ---> "$?
ln -s $INSTALLDIR/lib/gcc/arm-none-linux-gnueabi/4.7.2/libgcc.a $INSTALLDIR/lib/gcc/arm-none-linux-
gnueabi/4.7.2/libgcc_eh.a
ln -s $INSTALLDIR/lib/gcc/arm-none-linux-gnueabi/4.7.2/libgcc.a $INSTALLDIR/lib/gcc/arm-none-linux-
gnueabi/4.7.2/libgcc_s.a
echo -e " $VERT-Compilation en-tete eglibc réussi $NORMAL"
;;


9)
# eglibc
mkdir $BUILDDIR/eglibc
cd $BUILDDIR/eglibc
echo "libc_cv_forced_unwind=yes" > config.cache
echo "libc_cv_c_cleanup=yes" >> config.cache
../../sources/eglibc-2.16/libc/configure \
        --build=$BUILD \
        --host=$TARGET \
        --prefix=/usr \
        --with-headers=$SYSROOTDIR/usr/include \
        --config-cache \
        --enable-kernel=3.5.3 \
        --disable-profile --without-gd --without-cvs --enable-add-ons=ports,nptl --with-tls

echo -e "  $VERT Exit retourné $NORMAL ---> "$?
make -k install-headers cross_compiling=yes install_root=$SYSROOTDIR
echo -e "  $VERT Exit retourné $NORMAL ---> "$?
make -j$THREADS
echo -e "  $VERT Exit retourné $NORMAL ---> "$?
make install_root=$SYSROOTDIR install
echo -e "  $VERT Exit retourné $NORMAL ---> "$?
echo -e " $VERT-Compilation eglibc réussi $NORMAL"

;;



10)
unset CROSS
unset CC
unset LD
unset AR
unset AS

# GCC FINAL

mkdir $BUILDDIR/gcc
cd $BUILDDIR/gcc
export CC=gcc
echo "libc_cv_forced_unwind=yes" > config.cache
```

```
echo "libc_cv_c_cleanup=yes" >> config.cache
../../sources/$GCC/configure \
        --build=$BUILD \
        --target=$TARGET \
        --prefix=$INSTALLDIR \
        --with-sysroot=$SYSROOTDIR \
        --enable-languages=c \
        --with-float=soft \
        --disable-sjlj-exceptions \
        --disable-nls \
        --enable-threads=posix \
        --disable-libmudflap \
        --disable-libssp \
        --with-gnu-as \
        --with-gnu-ld \
        --disable-multilib \
        --enable-long-longx

echo -e "  $VERT Exit retourné $NORMAL ---> "$?
make all-gcc
echo -e "  $VERT Exit retourné $NORMAL ---> "$?
make install-gcc
echo -e "  $VERT Exit retourné $NORMAL ---> "$?
echo -e " $VERT-Compilation gcc réussi $NORMAL"

;;

11)
#-----------------------------------------#
#  Génération automatique d'un hello world  #
#-----------------------------------------#
cd
cd eGcross/
mkdir programme && cd programme
cat > hello_world_arm.c <<EOF

#include<stdio.h>

int main(void)
{

        printf("Hello from ARM \n");
        return 0;
}

EOF

arm-none-linux-gnueabi-gcc -o hello_world_arm hello_world_arm.c

;;


esac
done
```