

# TP1

## **Objectif :**

### **Configuration de BuildRoot :**

Je commence par charger la configuration de la sama5d3 du buildroot avec la commande :

```
> make atmel_sama5d3_xplained_defconfig
```

ensuite j'exécute la commande suivante afin de configurer plus en détail le système

```
> make menuconfig
```

Je configure le toolchain dans Toolchain, puis Toolchain Type → External Toolchain → Custom Toolchain

Puis dans Toolchain path je met le lien vers les binaires du Toolchain :

```
> /home/yoda/Desktop/racine/toolchain/armv7-eabihf—uclibc--stable-2020.02-2/bin/
```

Et dans toolchain prefix j'écris : arm-buildroot-linux-uclibcgnueabi-

je modifie les headers du toolchain en 4.4.x

Je modifie la version du toolchain de gcc en 8.x

Ensuite toujours dans toolchain, j'active :

```
> WCHAR Support
```

```
> locale support
```

```
> C++ Support
```

Puis OpenMP et SSP sont déjà désactivés, donc je les laisse désactivés.

Je vais dans System Configuration puis j'active /dev management (Dynamic using devtmpfs + mdev)

Quelques lignes en dessous dans la ligne Root password, je mets un mot de passe.

Il ne reste plus qu'à installer les packages suivants dans l'onglet Target Packages

Pour finir je décoche tout ce qui est contenu dans Kernel, et Bootloader et tout ce que je peux enlever dans Host Utilities

je quitte et sauvegarde le fichier config.

Il ne me reste plus qu'à compiler avec make

**Serveur NFS :**

Je décompresse le fichier rootfs.tar qui est contenu dans output/images dans le dossier nfsroot (bien sur je l'ai vidé avant)

Lorsque j'ai lancé la carte pour la première fois, j'ai eut des problème de permission, pour corriger cela, j'ai effectué cette commande dans le dossier nfsroot/bon : `sudo chown root * -R`

```
modprobe: can't change directory to '/lib/modules': No such file or directory
Initializing random number generator: OK
Saving random seed: random: dd: uninitialized urandom read (512 bytes read, 42 bits of entropy available)
OK
Starting network: ip: RTNETLINK answers: File exists
FAIL
Starting dropbear sshd: random: dropbear: uninitialized urandom read (32 bytes read, 49 bits of entropy available)
OK

Welcome to Buildroot
buildroot login: root
Password:
# echo nice
nice
#
```

ça fonctionne.

**Ajout d'application :**

Je remarque que la commande httpd, ne fonctionne pas, il faut donc que je l'ajoute a buildroot, pour faire cela, j'effectue un `make busybox-menuconfig`, et je vais activer la commande avec les lignes suivante :

- > make clean.
- > make busybox-menuconfig (ici dans Networking-Utilities → httpd)
- > make busybox
- > make -j12

Maintenant je peux vérifier que l'application httpd fonctionne.

**Ajout de package a Buildroot :**

je suis ce qui est écrit dans la doc, ainsi que dans le cours: <http://buildroot.uclibc.org/downloads/manual/manual.html#adding-packages>

Je commence par créer un dossier 'upload' dans buildroot/package

Ensuite dans buildroot/package/Config.in, j'ajoute un menu pour mes package.

```
menu "Text editors and viewers"
    source "package/ed/Config.in"
    source "package/joe/Config.in"
    source "package/less/Config.in"
    source "package/mc/Config.in"
    source "package/most/Config.in"
    source "package/nano/Config.in"
    source "package/uemacs/Config.in"
    source "package/vim/Config.in"
endmenu

menu "My Packages"
    source "package/upload/Config.in"
endmenu
```

→ ici

ensuite je crée un fichier Config.in :

```
config BR2_PACKAGE_UPLOAD
    bool "upload"
    default y
    help
        upload component.
```

Je met le fichier upload.c dans /package/upload/

Et je crée un fichier upload.mk :

```

/home/yoda/Desktop/buildroot-2020.02.11/package/upload/upload.mk - Mousepad
File Edit Search View Document Help
#####
#
# upload
#
#####

UPLOAD_VERSION = 1.0
UPLOAD_SOURCE = upload.c
UPLOAD_SITE = ~/Desktop/buildroot-2020.02.11/package/upload
UPLOAD_SITE_METHOD = local
UPLOAD_INSTALL_TARGET:=YES

define UPLOAD_BUILD_CMDS
$(MAKE) $(TARGET_CONFIGURE_OPTS) -C $(@)
endef

define UPLOAD_INSTALL_TARGET_CMDS
cp -r ~/Desktop/buildroot-2020.02.11/package/upload/www $(TARGET_DIR)
$(INSTALL) -D -m 0755 $(@)/upload $(TARGET_DIR)/www/cgi-bin/upload
endef

$(eval $(generic-package))

```

→ ici je copie l'arborescence du serveur dans rootfs

Puis je crée un fichier upload.hash qui contient le sha256 de upload.c

```

/home/yoda/Desktop/buildroot-2020.02.11/package/upload/upload.hash - Mousepad
File Edit Search View Document Help
c637cee3bbfb4b386e43298a59efb5357a0b7e0d075b03ff841f814d8202dbf4 upload.c

```

et un makefile :

```

/home/yoda/Desktop/buildroot-2020.02.11/pack - + x
File Edit Search View Document Help
CC=gcc
CFLAGS=

all: *.c
    $(CC) *.c -o upload

clean:
    rm -f upload.out upload
    rm -f *.o

```

Enfin je vais vérifier que le package upload est bien présent :

> make menuconfig

```

- Target packages - My Packages
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> sel
is selected [ ] feature is excluded

[ ] upload (NEW)

```

Et il est bien cocher !

Pour finir je copie le contenu du dossier www dans le dossier package/upload afin qu'elle soit copiée dans rootfs lors de la compilation.

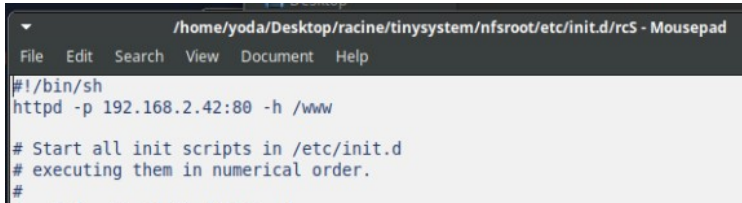
Afin de compiler j'exécute ces lignes :

- > make clean
- > make busybox-menuconfig (j'active Networking-Utilities → httpd)
- > make busybox
- > make -j12

Ensuite pour démarre le système il ne reste qu'as exécuter la commande suivante :

```
> httpd -p 192.168.1.42:80 -h /www
```

Pour finir afin que le serveur web se lance automatiquement au démarrage, j'ajoute cette ligne au fichier etc/init.d/rcS



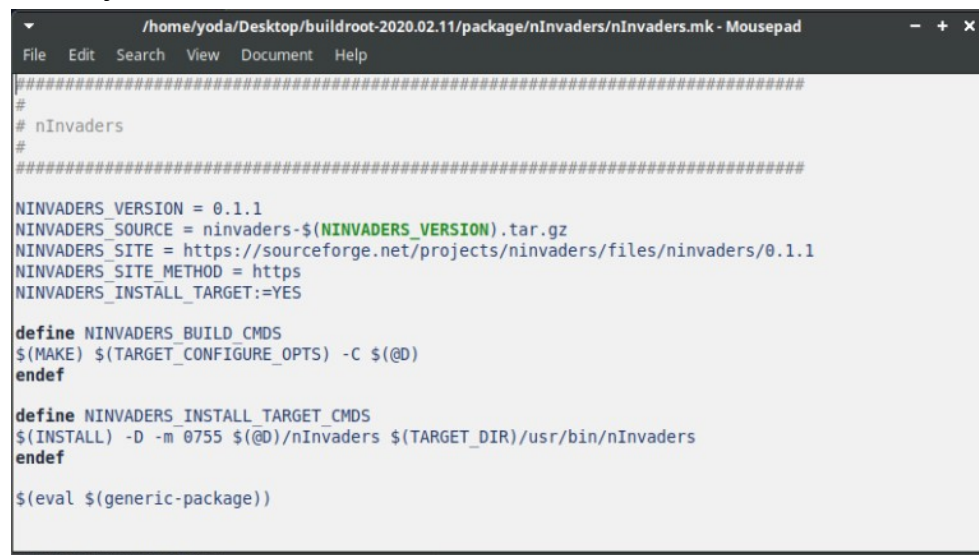
```
#!/bin/sh
httpd -p 192.168.2.42:80 -h /www

# Start all init scripts in /etc/init.d
# executing them in numerical order.
#
```

### **Ajout de nInvaders a buildroot :**

Je fais une copie du dossier package/upload et le renomme en nInvaders (ainsi que toutes les apparition du mot upload).

Ensuite je modifie le fichier Ninvaders.mk



```
#####
#
# nInvaders
#
#####

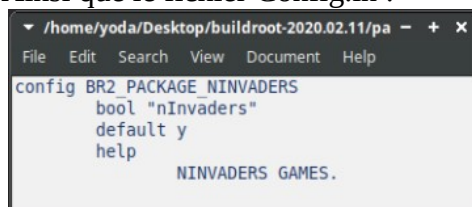
NINVADERS_VERSION = 0.1.1
NINVADERS_SOURCE = ninvaders-$(NINVADERS_VERSION).tar.gz
NINVADERS_SITE = https://sourceforge.net/projects/ninvaders/files/ninvaders/0.1.1
NINVADERS_SITE_METHOD = https
NINVADERS_INSTALL_TARGET:=YES

define NINVADERS_BUILD_CMDS
$(MAKE) $(TARGET_CONFIGURE_OPTS) -C $(@D)
endef

define NINVADERS_INSTALL_TARGET_CMDS
$(INSTALL) -D -m 0755 $(@D)/nInvaders $(TARGET_DIR)/usr/bin/nInvaders
endef

$(eval $(generic-package))
```

Ainsi que le fichier Config.in :



```
config BR2_PACKAGE_NINVADERS
bool "nInvaders"
default y
help
    NINVADERS GAMES.
```

et je colle le patch dans package/nInvaders/0.1.1-1.patch :

```

/home/yoda/Desktop/buildroot-2020.02.11/package/nInvaders/0.1.1-1.patch - Mousepad
File Edit Search View Document Help
diff -u Original/ninvaders-0.1.1/aliens.c ninvaders-0.1.1/aliens.c
--- Original/ninvaders-0.1.1/aliens.c      2003-05-08 21:19:50.000000000 +0200
+++ ninvaders-0.1.1/aliens.c      2021-01-30 13:32:26.474787032 +0100
@@ -163,7 +163,7 @@
     aliens.right=-1;
     aliens.bottom=-1;
     shipnum=0;
-    for (k=0;k<11;k++) {
+    for (k=0;k<ALIENS_MAX_NUMBER_X;k++) {
         lowest_ship[k]=-1;
     }

diff -u Original/ninvaders-0.1.1/Makefile ninvaders-0.1.1/Makefile
--- Original/ninvaders-0.1.1/Makefile      2003-05-08 21:19:50.000000000 +0200
+++ ninvaders-0.1.1/Makefile      2021-01-16 14:34:54.000000000 +0100
@@ -1,6 +1,6 @@
-CC=gcc
+CC=arm-buildroot-linux-uclibcgnueabi-gcc
CFLAGS=-O3 -Wall
-LIBS=-lcurses
+LIBS=-lcurses -I/home/yoda/Desktop/tmp_ncurse/usr/include -L/home/yoda/Desktop/tmp_ncurse/usr/lib

CFILES=globals.c view.c aliens.c ufo.c player.c nInvaders.c
HFILES=globals.h view.h aliens.h ufo.h player.h nInvaders.h
@@ -11,6 +11,6 @@
    $(CC) $(LDLAGS) -o$@ $(OFILES) $(LIBS)

.c.o:
-    $(CC) -c -I. $(CFLAGS) $(OPTIONS) $<
+    $(CC) -c -I. $(CFLAGS) $(OPTIONS) $(LIBS) $<
clean:
    rm -f nInvaders $(OFILES)|

```

### Mise en place de la gestion de la clé USB :

Je commence par créer un fichier un fichier correct\_mdef.patch, qui permette de corriger mdev. Et je le met dans buildroot/skeleton/etc

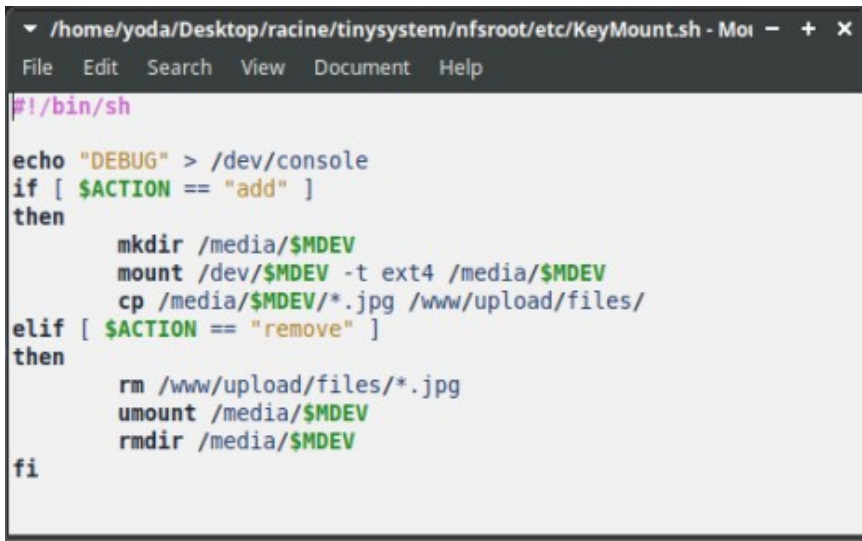
```

/home/yoda/Desktop/buildroot-2020.02.11/system/skeleton/etc/cc - + x
File Edit Search View Document Help
--- mdev_orig.conf      2021-03-10 16:46:07.694156814 +0100
+++ mdev.conf      2021-03-10 16:45:15.878047139 +0100
@@ -37,3 +37,4 @@

# load modules
$MODALIAS=.* root:root 660 @modprobe "$MODALIAS"
+sd[a-z][0-9]+ root:root 660 */etc/KeyMount.sh

```

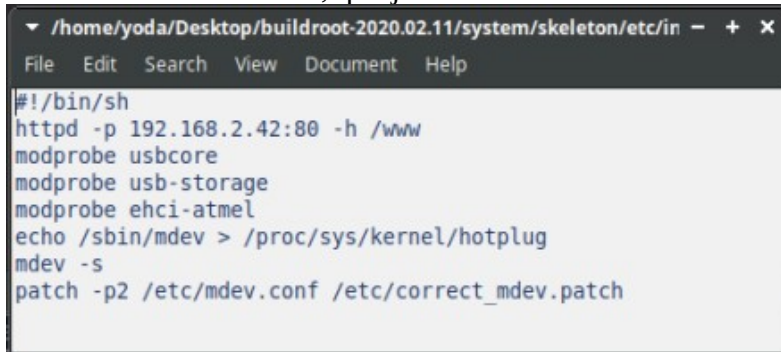
Ensuite je crée un fichier KeyMount.sh, que je met aussi dans buildroot/skeleton/etc



```
#!/bin/sh

echo "DEBUG" > /dev/console
if [ $ACTION == "add" ]
then
    mkdir /media/$MDEV
    mount /dev/$MDEV -t ext4 /media/$MDEV
    cp /media/$MDEV/*.jpg /www/upload/files/
elif [ $ACTION == "remove" ]
then
    rm /www/upload/files/*.jpg
    umount /media/$MDEV
    rmdir /media/$MDEV
fi
```

Je crée un fichier S01rcS, que je met dans :buildroot/skeleton/etc/init.d/



```
#!/bin/sh
httpd -p 192.168.2.42:80 -h /www
modprobe usbcore
modprobe usb-storage
modprobe ehci-atmel
echo /sbin/mdev > /proc/sys/kernel/hotplug
mdev -s
patch -p2 /etc/mdev.conf /etc/correct_mdev.patch
```

Je copie les module qui me permette de géré les clé USB (usbcore) que nous avons crée lors d'un précédent TP dans system/skeleton/lib/modules

Puis je recompile le tout

- > make clean
- > make busybox-menuconfig (j'active Networking-Utilities → httpd)
- > make busybox
- > make -j12

Je décompresse tout le contenu du fichier /output/images/rootfs.tar dans le dossier nfsroot.  
J'exécute la commande : `sudo chown root * -R` afin de donner les droit au système embarqué.

Pour finir je lance le système embarqué, et tout fonctionne.