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'execute la commande suivante afin de configuré plus en detail le systeme
> 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 binaire du Toolchain :
> /home/yoda/Desktop/racine/toolchain/armv7-eabihf—uclibc--stable-2020.02-2/bin/

Et dans toolchain prefix j'ecris : arm-buildroot-linux-uclibcgnueabihf-
je modifie les header 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 desactivé, donc je les laisse desactivé.
Je vais dans System Configuration puis j'active /dev management (Dynamic using devtmpfs + mdev)

Quelque ligne en dessous dans la ligne Root password, je met un mot de passe.

Il ne reste plus qu'as installer les package suivant dans l'onglet Target Packages

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

je quitte et sauvegarde le fichier config.

Il ne me reste plus qu'as 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 premiere fois, j'ai eut des probleme 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**

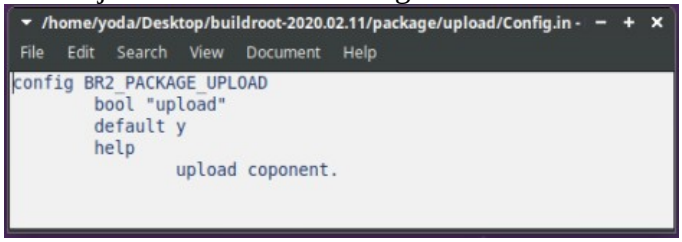
Je commence par crée un dossier 'upload' dans buildroot/package
Ensuite dans buildroot/package/Config.in, j'ajoute un menu pour mes package.

```
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
```

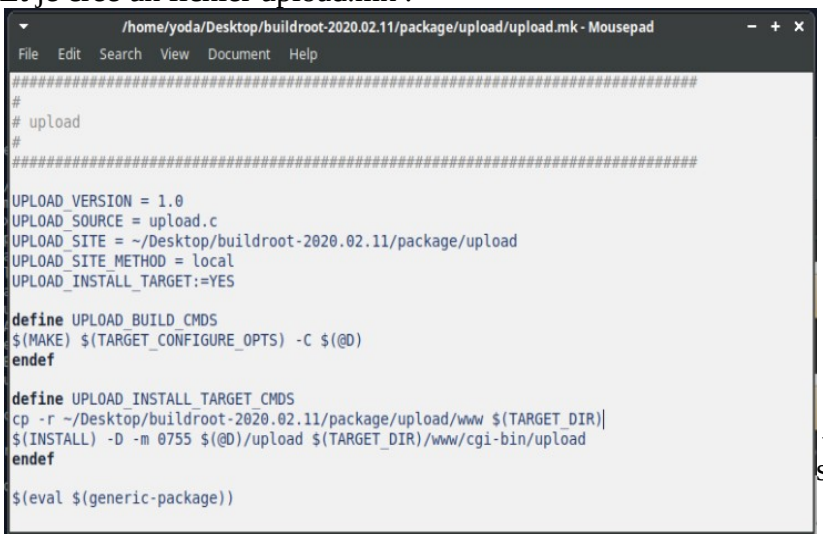
→ ici

ensuite je crée un fichier Config.in :



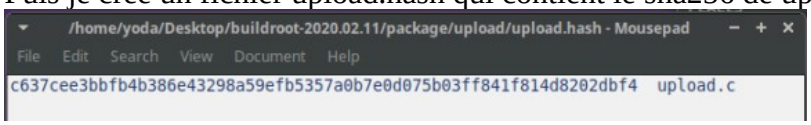
Je met le fichier upload.c dans /package/upload/

Et je crée un fichier upload.mk :

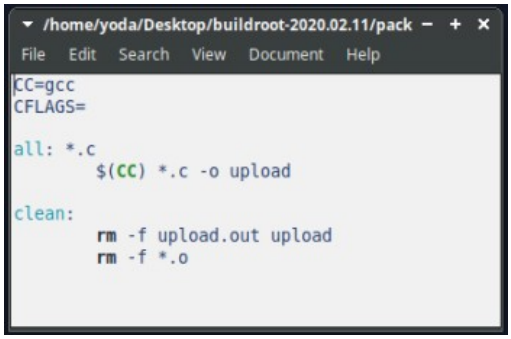


→ ici je copie l'arborecence du serveur dans rootfs

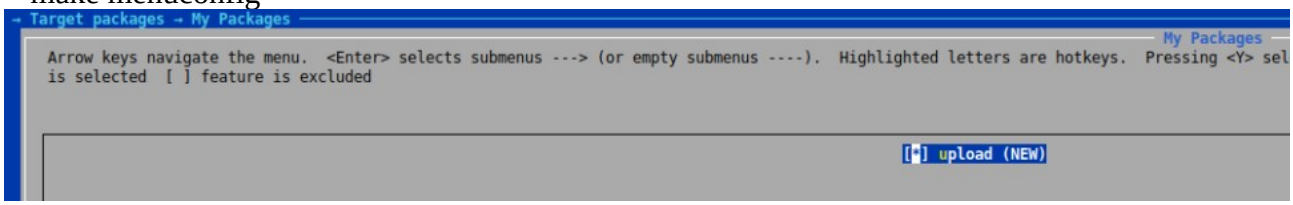
Puis je crée un fichier upload.hash qui contient le sha256 de upload.c



et un makefile :



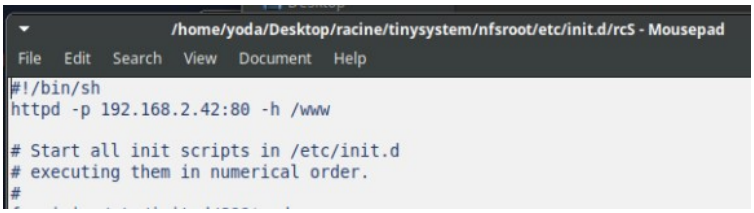
Enfin je vais vérifier que le package upload est bien présent :
> make menuconfig



Et il est bien cocher !
Pour finir je copie le contenu du dossier www dans le dossier package/upload afin quelle soit copié 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

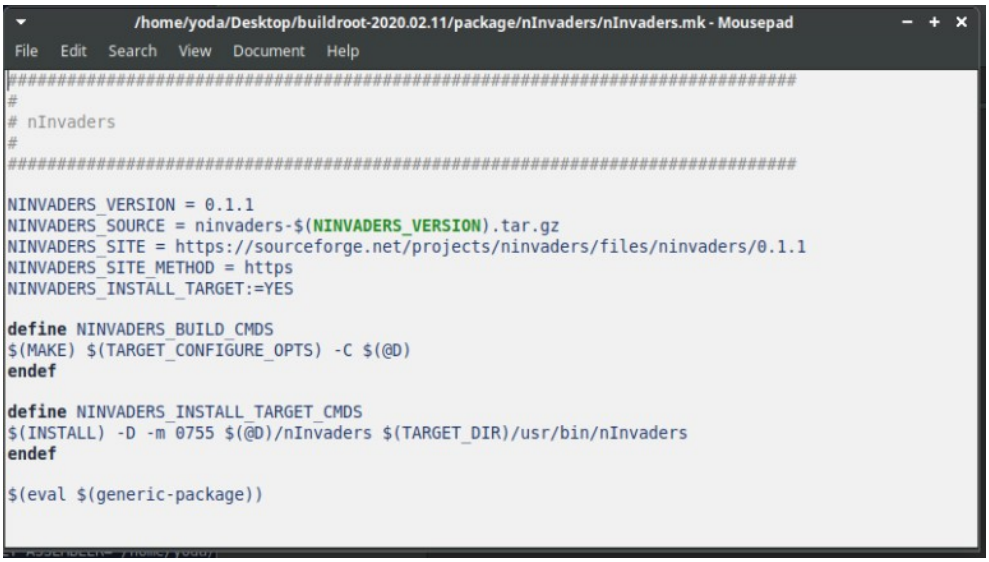
Ensuite pour démarre le système il ne reste qu'as exécuter la commande suivante :
> httpd -p 192.168.1.42:80 -h /www

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

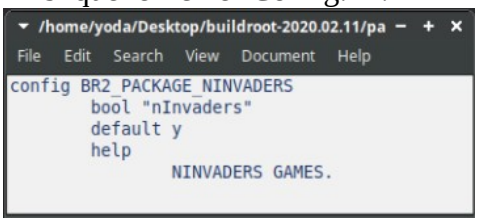


Ajout de nInvaders a buildroot :

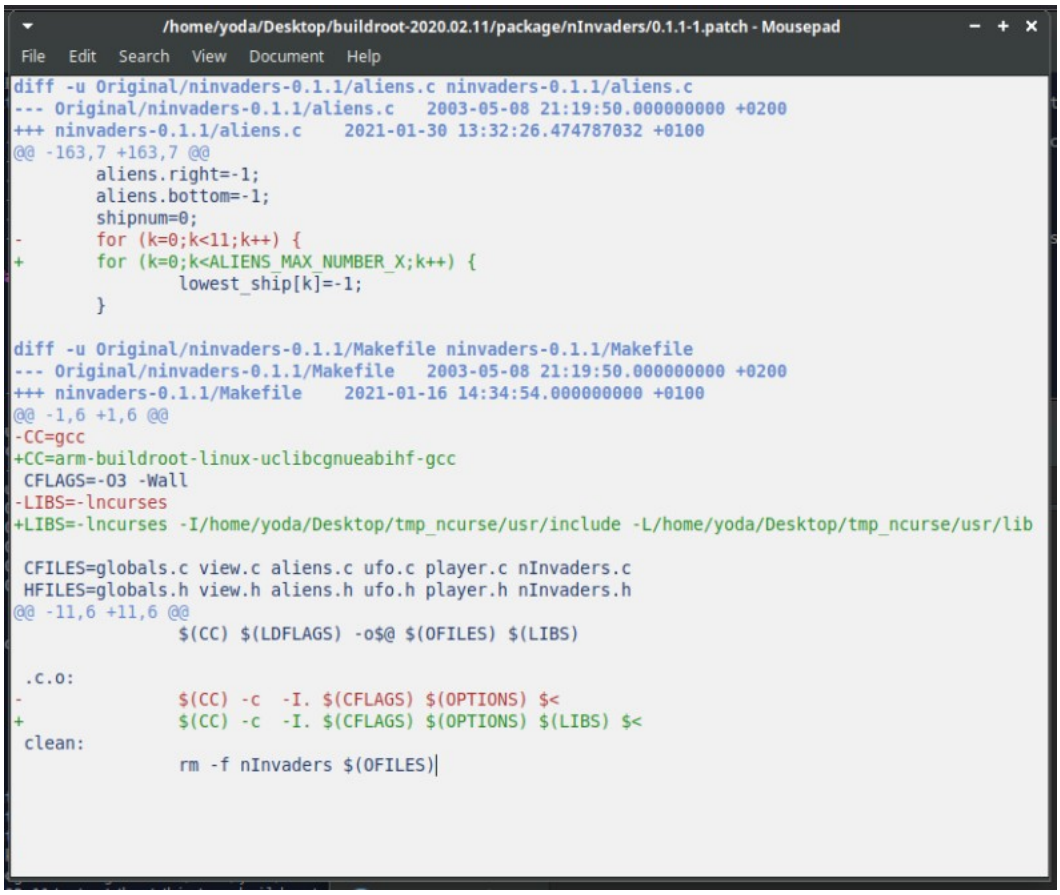
Je fais une copie du dossier package/upload et le renomme en nInvaders (ainsi que toutes les apparition du mot upload).
Ensuite je modifie le fichier Ninvaders.mk



Ainsi que le fichier Config.in :

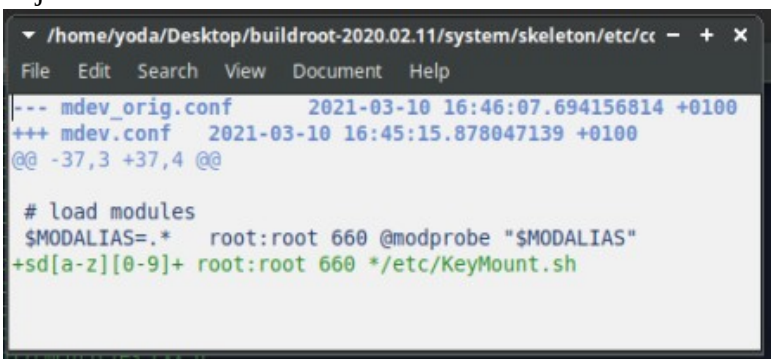


et je colle le patch dans package/nInvaders/0.1.1-1.patch :

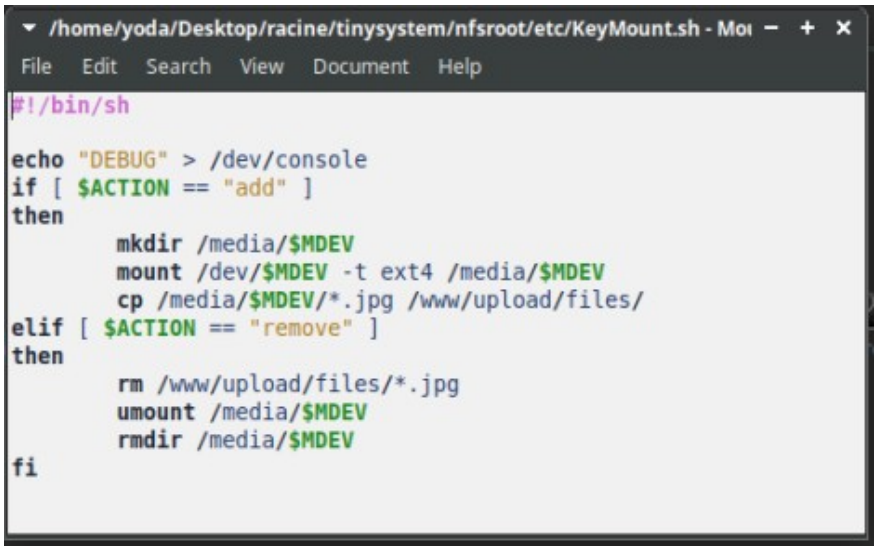


Mise en place de la gestion de la clé USB :

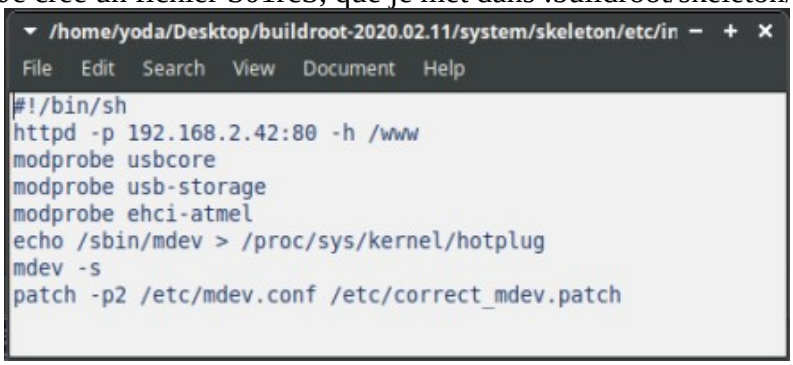
Je commence par crée un fichier un fichier correct_mdef.patch, qui permettre de corriger mdev.
Et je le met dans buildroot/skeleton/etc



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



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



Je copie les module qui me permette de géré les clé USB (usbcore) que nous avions 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.