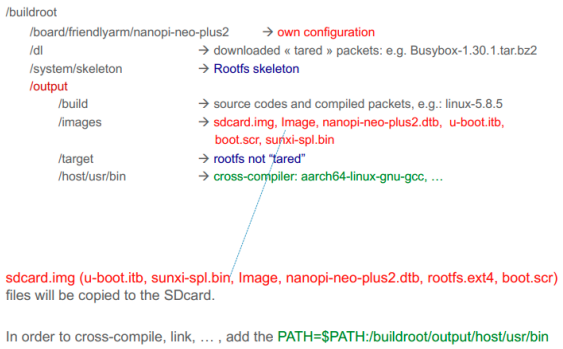
1 Buildroot

1.1 Principaux répértoires



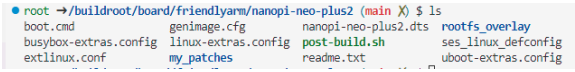
Ce qui est manquant dans le répértoire dans le dossier output sera recompilé lors de la commande `make` ou en précisant le paquet : `make <package>-rebuild` à refaire.

1.2 Principe de fonctionnement

Basé sur des fichiers `make kconfig`, buildroot est un outil qui automatise le process de construction d'un système Linux embarqué en utilisant la cross-compilation. Lors de la commande `make` il va s'occuper de compiler l'entier du système et préparer une image complète et prête pour l'utilisation.

1.3 Configuration pour un hardware donné

Il y a un répertoire situé dans `board/friendlyarm/nanopi-neo-plus2` qui contient plusieurs fichiers intéressants. On y retrouve notamment le fichier `nanopi-neo-plus2/nanopi-neo-plus2.dts` qui contient le FTD ("Flattened Device Tree") qui décrit le hardware. On indique donc à buildroot l'emplacement de ce fichier avec la commande `menuconfig`. Autres fichiers intéressants :



1.4 Patch buildroot

Il faut spécifier le dossier des patchs `make menuconfig` nous on a fait dans `/nanopi-neo-plus2/my_patches` et un dossier par package à patcher. Ensuite la technique consiste à profiter de l'outil git et se mettre dans `/buildroot/output/build/uboot-2020.07` et de faire:

```
git init --initial-branch=main
git add .
cd common/
git add .
git commit -m "1st commit"
```

Faire ensuite toutes les modifications à faire, les stager et faire la commande qui créé le patch au format voulu par buildroot:

```
git diff --cached --stat -p > /buildroot/boa⌋
↪ rd/friendlyarm/nanopi-neo-plus2/my_patch⌋
↪ es/uboot/0001_stack_protector.patch
```

On peut ensuite supprimer le paquet en question dans build et il se téléchargera et patchera automatiquement au prochain `make`.

1.5 Configuration de buildroot, u-boot, kernel

Pour buildroot, on utilise `make menuconfig` ça se sauvegarde dans `.config` ou dans `configs/ses_defconfig` (seulement les changement sont dans ce fichier, qui est à la base une copy de `friendlyarm_nanopi_neo_plus2_defconfig` qui se situe au même endroit).
Pour u-boot, on va dans `make menuconfig` catégorie bootloaders et ça va se sauver dans

```
/buildroot/output/build/uboot-xx/configs/nan⌋
↪ opi_neo_plus_defconfig
```

Et dans un fragment files qui ne contient que des petites modifications (save old, make new, use diff, add it to this fragment file):

```
/buildroot/board/friedlyarm/nanopi-neo-plus/⌋
↪ uboot-extras.config
```

Pour le kernel, `make linux-menuconfig`, ça save dans `/buildroot/output/build/linux-xx/defconfig` et nous on le copie et le met dans `ses_linux_defconfig`

1.6 Génération de la carte SD

Pour créer le fichier `sdcard.img` utilise le script `genimage.sh` qui va aller aussi utiliser le fichier `nanopi-neo-plus2/genimage.cfg` qui spécifie les différentes partitions à créer et les images (venant de `output/images`) à mettre dedans.

1.7 Génération du rootfs

Il est copié depuis `/buildroot/system/skeleton` pour le mettre dans `/buildroot/output/target`. Il est ensuite possible d'ajouter des fichiers et des répértoires avec le `/nanopi-neo-plus2/rootfs_overlay`. Après la commande `make` le pseudo rootfs est créé et placé dans un des fichiers images de sorties `rootfs.xxx`

1.8 rootfs_overlay

On peut préprarer la structure des fichiers de la cible dans ce dossier

1.9 Installer un nouveau package dans buildroot

Pour utiliser un package, il faut faire `make menuconfig` et aller choisir le package à aller utiliser. Sinon il faut ajouter le nouveau package "foo" dans le dossier `/buildroot/package`. Il doit contenir au moins:

- Config.in: in kconfig language : on pourra y accéder à travers `make menuconfig`
- foo.mk: fichier qui décrit ou prendre les sources et comment les installer
- optional foo.hash: pour vérifier l'intégrité des fichiers à télécharger
- optional Sxx_foo: le startup script pour le package foo