Yocto - devtool - et autres...
Comment développer efficacement sous avec Yocto

Mickaël Tansorier

Présentation des outils et commandes pratique pour travailler sous Yocto

Objectif de la présentation

- Présenter rapidement Yocto
- Présenter des outils utiles
  - `devtool`
  - `opkg`
  - `...`

Plan

Présentation de Yocto

Présentation de Yocto

D'où vient ce nom ?

Definition

Yocto est un préfixe représentant 10^{-24} unités (SI)

Qu'est ce qu'est vraiment Yocto ?

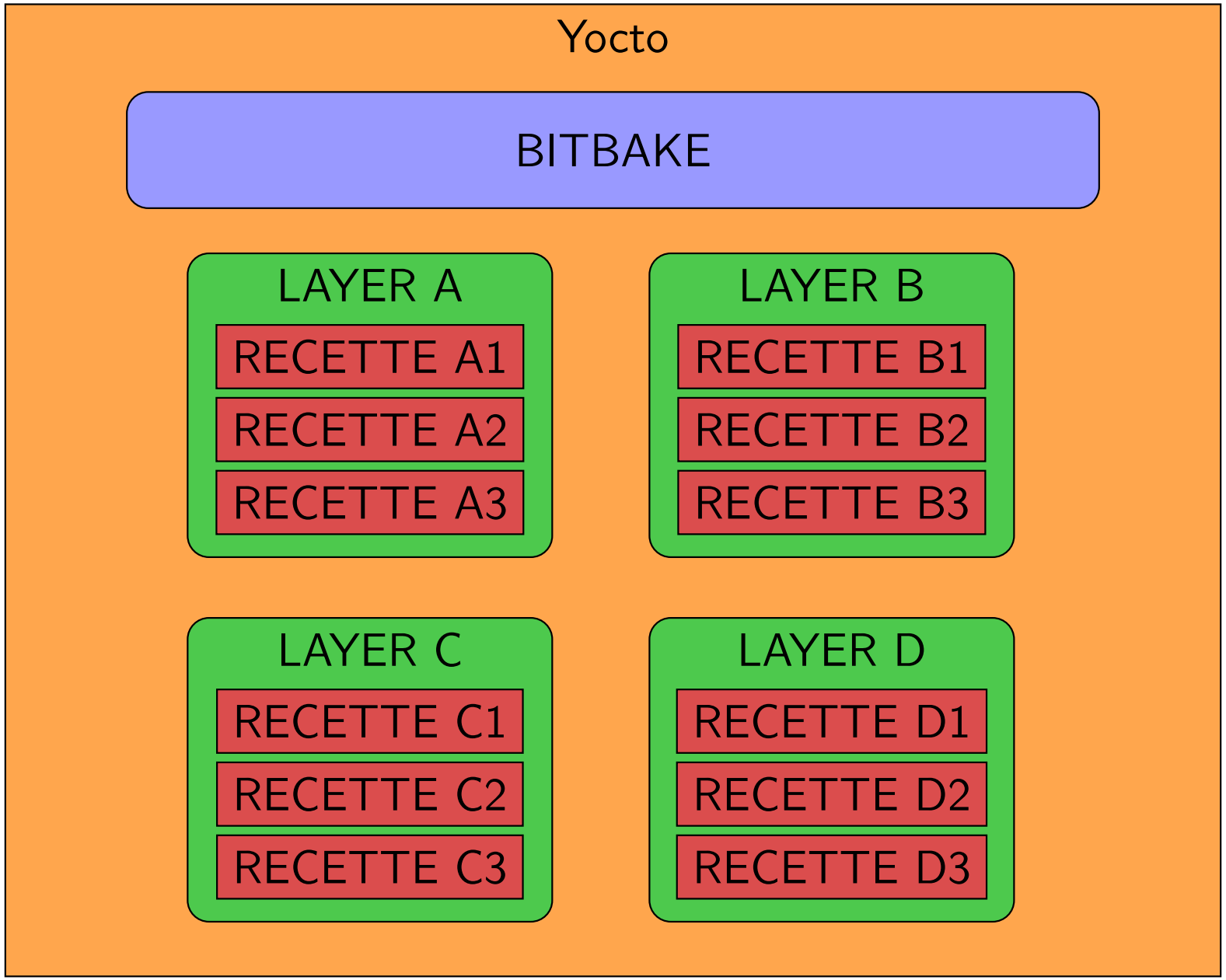
Yocto est un outil qui répond au besoin de générer une distribution **Linux embarqué** pour un matériel **dédié**.

Yocto utilise le principe de `MACHINE` et de `DISTRO` qui sent à la différenciation de l'**architecture matérielle** de l'**application logicielle** de la cible

- `MACHINE` : définit l'architecture matérielle
- `DISTRO` : définit la distribution à générer

Il dispose de plusieurs outils très pratique pour le développement :

- `devtool`
- `ipk/opkg`


Yocto
BITBAKE
LAYER A
RECETTE A1
RECETTE A2
RECETTE A3
LAYER B
RECETTE B1
RECETTE B2
RECETTE B3
LAYER C
RECETTE C1
RECETTE C2
RECETTE C3
LAYER D
RECETTE D1
RECETTE D2
RECETTE D3

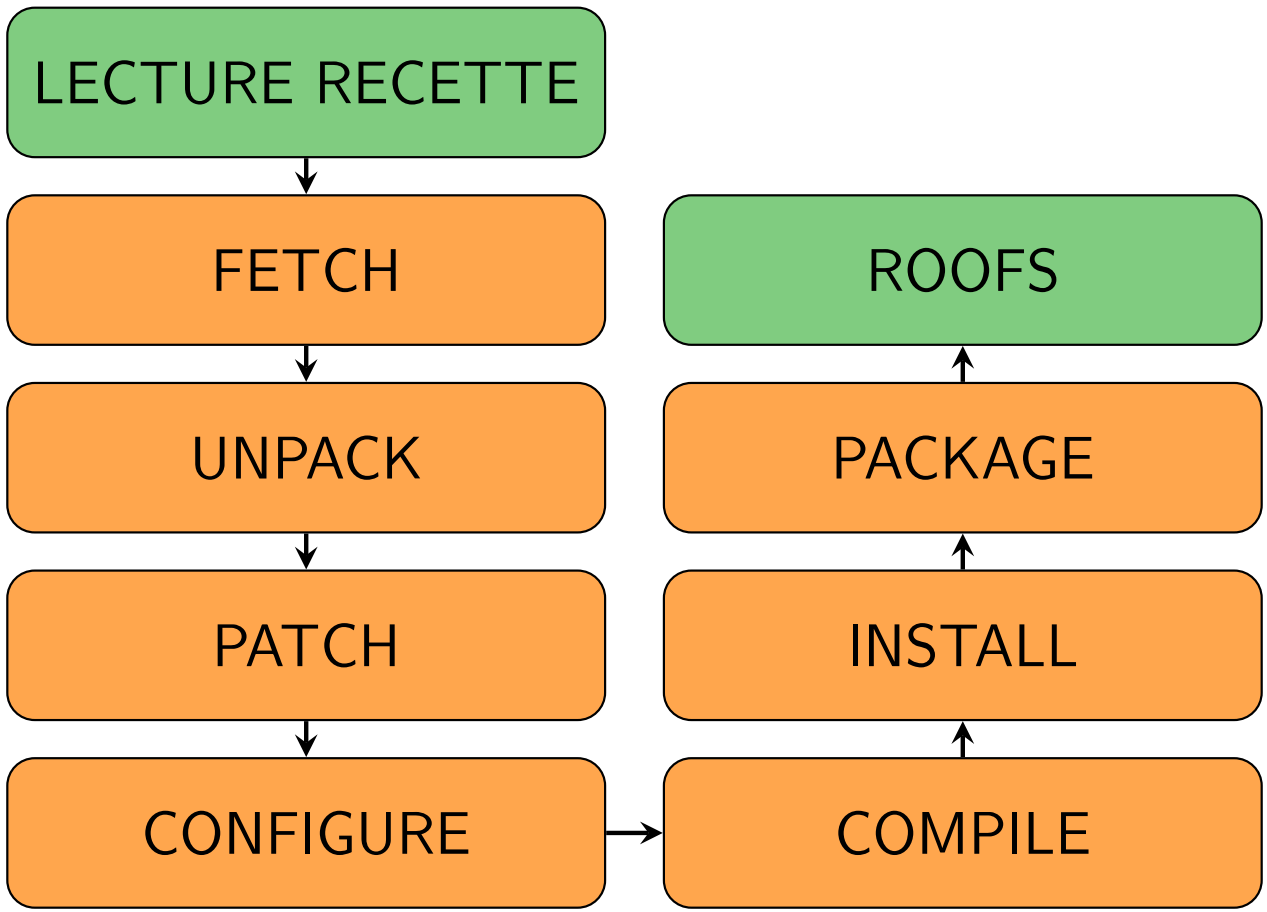
Avant de passer aux recettes, qui fait le travail dans Yocto ?

bitbake

bitbake c'est quoi ?

- Un moteur d'exécution de tâches écrite en Python
- Fonctionne en ligne de commande
- Exécute automatiquement les tâches nécessaires à la fabrication de la cible fournie

bitbake



Pour avoir les vrais étapes d'une recette : `bitbake <recette> -c listtasks`

recette

À quoi ça ressemble une recette ?

```
DESCRIPTION = "Exemple d'une description d'une recette"
LICENSE = "MIT"
LIC_FILES_CHKSUM = "file://${COMMON_LICENSE_DIR}/MIT;md5=0835ade698e0bcf8506ecda2f7b4f302"

SRC_URI = "git://github.com/exemple/exemple.git \
           file://un-patch.patch \
           file://un-fichier \
"

SRCREV = "2693ca21cee8a729d74682fd86a4818f2b050228"

S = "${WORKDIR}/git"

do_configure() {
    # Commentaire d'une fonction pour spécifier la config
}

EXTRA_OECONF = "--une-option"

do_install() {
    install -d ${D}${bindir}/un-dossier
    install -m 0755 ${S}/un-binaire-construit ${D}${bindir}/un-dossier/un-binaire-construit
}

FILES_${PN} = "${bindir}/un-dossier/un-binaire-construit"
```

devtool

devtool

Exemple de l'utilisation de l'outil devtool

`devtool` est un outils très utiles lorsque l'on souhaite créer, développer ou modifier une recette et ses sources.

Les commandes de base :

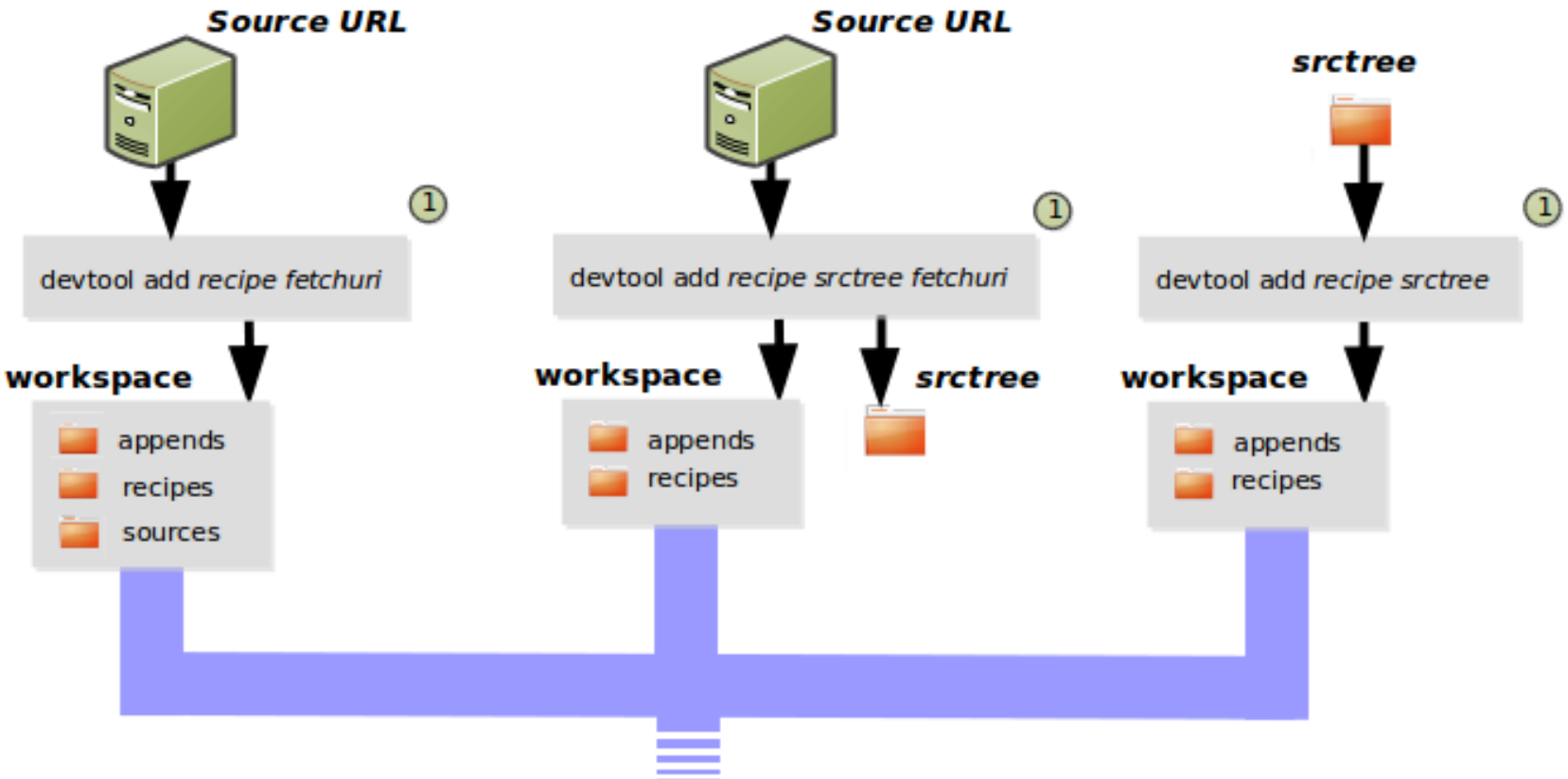
`devtool add` Ajoute un nouveau software à construire

`devtool modify` Génère un environnement pour modifier les sources d'une recette

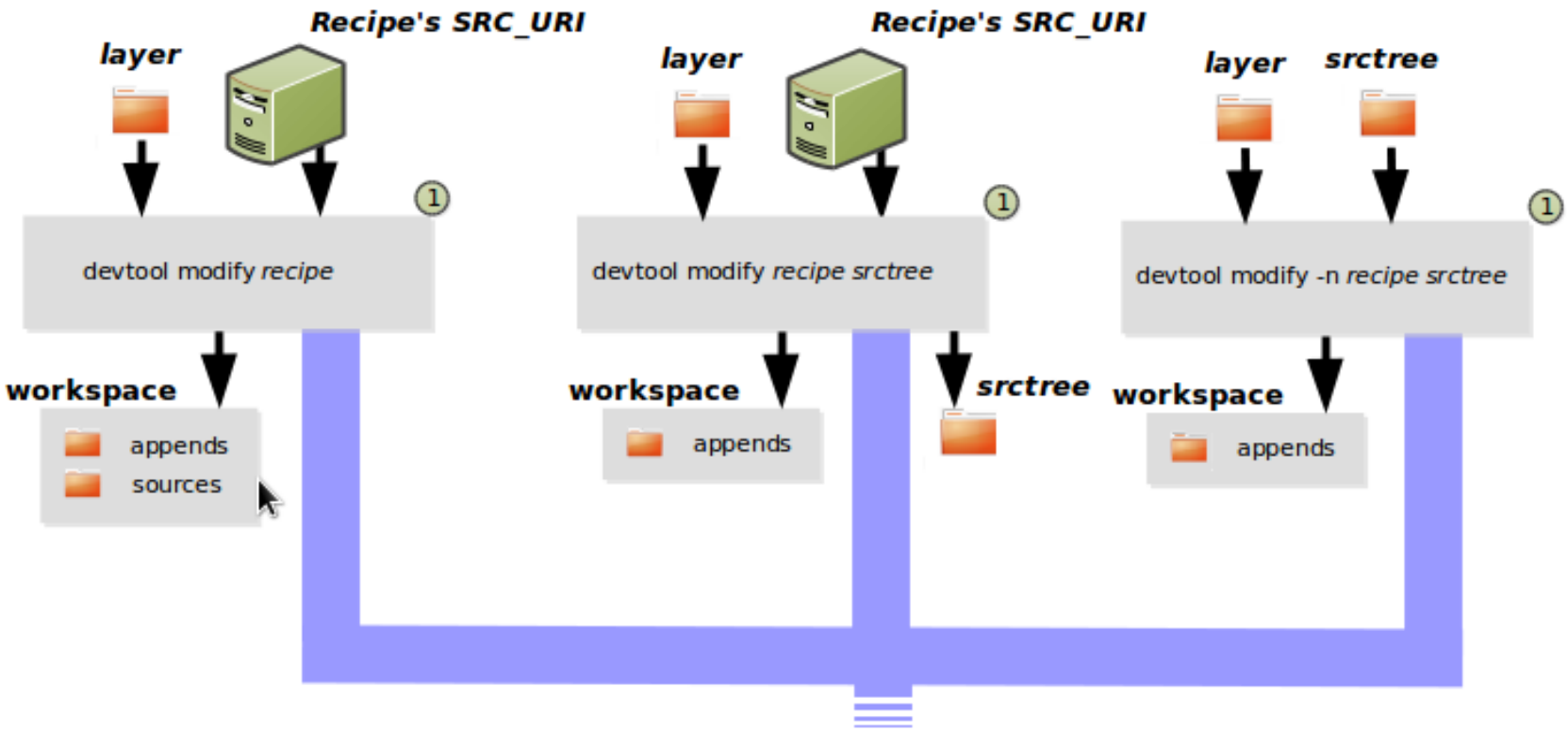
`devtool upgrade` Met à jour une recette existante

`devtool reset` Arrête le contexte de développement

Les sources peuvent provenir de plusieurs endroits différent



De même pour modifier une recette



Dès lors devtool créer un layer spécial (workspace) qui prend la priorité maximal sur les autres layers.

```
$ bitbake-layers show-layers
NOTE: Starting bitbake server...
layer                 path                                      priority
==========================================================================
meta                  /home/[...]/meta                          5
meta-poky             /home/[...]/meta-poky                     5
meta-yocto-bsp        /home/[...]/meta-yocto-bsp                5
workspace             /home/[...]/build/workspace               99
meta-raspberrypi      /home/[...]/build/../meta-raspberrypi     9
meta-python           /home/[...]/build/../meta-openembedded/meta-python  7
meta-oe               /home/[...]/build/../meta-openembedded/meta-oe  6
meta-meetup           /home/[...]/build/../meta-meetup          10
```

Dans ce layer on retrouve

- les sources mis sous git et patché
- un bbappend de la recette

```
$ cd $POKY/build/workspace/
$ tree -L 2
.
├── appends
│   └── weston_2.0.0.bbappend
├── conf
│   └── layer.conf
├── README
└── sources
    └── weston
```

Exemple pratique avec la recette `weston`

Modification avec devtool des sources de weston

```
$ devtool modify weston
$ cd $POKY/build/workspace/sources/weston/
$ vim libweston/compositor-wayland.c +1655
```

Ajout du patch "Fix an uninitialized variable"

```
@@ -1652,6 +1652,7 @@ input_handle_axis(void *data,
    struct wl_pointer *pointer,

     weston_event.axis = axis;
     weston_event.value = wl_fixed_to_double(value);
+   weston_event.has_discrete = false;

     if (axis == WL_POINTER_AXIS_VERTICAL_SCROLL &&
         input->vert.has_discrete) {
```

Les étapes :

1. Faire la modification
2. Tester normalement
3. Commiter
4. Appliquer la modification sous forme de patch

```
$ devtool update-recipe weston
[...]
NOTE: Adding new patch 0001-Fix-an-uninitialized-variable.patch
NOTE: Updating recipe weston_2.0.0.bb
```

5. Ajouter la modification dans son layer
6. Arrêter devtool

```
$ devtool reset weston
```

Dans le cas où vous modifiez déjà une recette avec un bbappend, je vous conseil d'extraire les patchs à la main :

```
$ cd $POKY/build/workspace/sources/weston/
$ git format patch -<nb_patch>
```

Il ne reste plus qu'a copier ces patch dans la recette et d'ajouter dans le bbappend avec :

```
FILESEXTRAPATHS_prepend := "${THISDIR}/${PN}:"
SRC_URI += "file://000X-mon-patch.patch"
```

Exemple pour deployer directement ses modifications sur cible

Après modification de source avec devtool il est possible de déployer ces modification directement sur une cible avec :

```
$ devtool deploy-target <recette> <user@cible>
```

La cible doit disposer d'un serveur ssh.

Attention, devtool n'envoit pas un paquet mais le résultat de la recette, donc pas besoin d'un `package-management` sur cible.
Une fois votre mise au point terminée vous pouvez supprimer tous les fichiers ajoutés par votre recette la commande :

```
$ devtool undeploy-target <recette> <user@cible>
```

Exemple de l'utilisation des l'outils opkg et ipk

opkg est un gestionnaire de paquets

opkg est gestionnaire de paquet tout comme apt l'est pour les distribution hérité de Debian.

Les commandes de base :

`opkg update` Mettre à jour la list des paquets

`opkg upgrade <pkgs>` Mettre à jour un paquet

`opkg install <pkgs|url>` Installer un paquet

`opkg remove <pkgs>` Supprimer un paquet

Créer son propre serveur opkg

Pour créer son serveur opkg il suffit d'avoir un sereur http.
On peut le crée simplement avec python en remplaçant les variable par leur valeur :

```
$ cd ${POKY}/build/tmp/deploy/ipk
$ ${NATIVE_SDK_BINS}/usr/bin/python-native/python -m
    SimpleHTTPServer ${IPK_SERVER_PORT}
```

Avec python3 :

```
$ cd ${POKY}/build/tmp/deploy/ipk
$ ${NATIVE_SDK_BINS}/usr/bin/python-native/python3 -m http.server ${
    IPK_SERVER_PORT}
```

Définir le serveur opkg sur cible

Ajouter ces lignes dans le fichier `opkg.conf` en remplaçant les variable correspondante par leur valeur :

```
src/gz all http://ipk-server:${IPK_SERVER_PORT}/all
src/gz ${MACHINE_ARCH} http://ipk-server:${IPK_SERVER_PORT}/${MACHINE_ARCH}
src/gz ${MACHINE_SOCARCH} http://ipk-server:${IPK_SERVER_PORT}/${
    MACHINE_SOCARCH}
src/gz ${TUNE_PKGARCH} http://ipk-server:${IPK_SERVER_PORT}/${TUNE_PKGARCH}
```

Recette opkg

Ajouter dans `/etc/hosts` le serveur :

```
${IP_HOST}      ipk-server
```

Recette netbase

Quelques outils et commandes diverses qui peuvent intéresser

environnement

Pour connaitre l'état de n'importe quelle recette et quelle variable est compléter par quel fichier de recette, l'option -e est le plus puissant.

```
$ bitbake <recipe> -e
```

La sortie est assez velue, il vaut mieux l'ouvrir dans un éditeur comme :

```
$ bitbake <recipe> -e | vim -
```

environnement

Si vous souhaitez tester des commandes dans un environnement de cross compilation, l'option `devshell` est l'outils qu'il faut :

```
$ bitbake <recipe> -c devshell
```

defconfig

Ajouter un fichier de configuration.

Pour la plupart des paquets la gestion de yocto reconnais les defconfig. Il faut ajouter dans un bbappend pour dans une recette maitrisé :

```
FILESEXTRAPATHS:prepend := "${THISDIR}/${PN}:"
SRC_URI += "file://defconfig"
```

savedefconfig

Ajouter une option de configuration à un paquet facilement.

Ceci fonctionne si vous avez le defconfig dans votre layer.

1. Ajouter une config

```
$ bitbake linux-fslc -c menuconfig
```

2. Tester
3. Générer le nouveau defconfig

```
$ bitbake linux-fslc -c savedefconfig
```

4. Remplacer le defconfig par sa génération

diffconfig

Ajouter une option de configuration à un paquet facilement.

1. Ajouter une config

```
$ bitbake linux-fslc -c menuconfig
```

2. Tester
3. Extraire la configuration et ses dépendances mutualisé

```
$ bitbake linux-fslc -c diffconfig
linux-fslc-4.10-r0 do_diffconfig: Config fragment has been
     dumped into:
 /home/user/yocto/build/tmp/work/archi/linux-fslc/4.10-r0/
      fragment.cfg
```

4. Ajouter le fragment à la recette

```
FILESEXTRAPATHS_prepend := "${THISDIR}/${PN}:"
SRC_URI += "file://myfragment.cfg"
```

Après la génération du diffconfig il faut ajouter l'option à la recette car elle est retiré du config

Compilation

Si vous souhaitez compiler une image le plus loin possible après une erreur :

```
$ bitbake <image> -k
```

Quel parquet embarque quoi comme ficher

Lister les fichier contenue dans un paquet :

```
$ oe-pkgdata-util list-pkg-files <recipe>
```

Trouver quel paquet fournis quel fichier :

```
$ oe-pkgdata-util find-path /etc/opkg/pokg.conf
```

buildhistory

Buildhistory fourni un ensemble de détails sur le contenue des paquets, leur dépendances, et leur inclusion dans une image.

Ajouter dans `conf/local.conf` :

```
INHERIT += "buildhistory"
BUILDHISTORY_COMMIT = "1"
```

Le résultat se situe dans le dossier de build dans `buildhistory`.

Autocompletion

Pour faciliter le développement l'autocompletion est plus confortable.

Il existe des projets, mais qui date un peut...
`https://github.com/sergioprado/bitbake-bash-completion`
`https://github.com/lukaszgard/bitbake-completion`

Il faut copier les fichier dans :

```
/etc/bash_completion.d/
```

Suivez les instructions des dépôts.

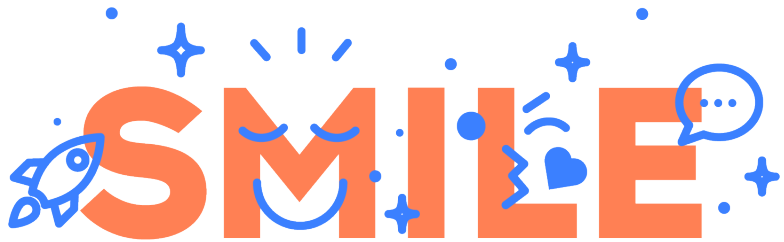
Sources

Ce document à été rédigé à partir des sources suivantes :

- `www.yoctoproject.org`
- `www.linuxembedded.fr`
- `https://openwrt.org/docs/guide-user/additional-software/opkg`

Merci de votre attention !

Quetions ?



Mickaël Tansorier
mickael.tansorier@smile.fr
mickael@tansorier.fr