www.ciose.fr

2015 SophiaConf
Le cycle azuréen de conférences Open Source

CIO SYSTEMES EMBARQUES
SOLUTIONS INFORMATIQUES ET ELECTRONIQUES

Concepts et avantages des outils proposés par Yocto

C. Charreyre
christian.charreyre@ciose.fr
http://www.ciose.fr
http://fr.slideshare.net/charreyre
@CIOinfoindus

www.ciose.fr

Licence

Attribution-Noncommercial-Share Alike 4.0 International

- You are free:
  - to Share - copy and redistribute the material in any medium or format
  - to Adapt - remix, transform, and build upon the material

  The licensor cannot revoke these freedoms as long as you follow the license terms.
- Under the following conditions:
  - Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
  - NonCommercial — You may not use the material for commercial purposes.
  - ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
- No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.
- License text : http://creativecommons.org/licenses/by-nc-sa/4.0/legalcode

2015 SophiaConf
Le cycle azuréen de conférences Open Source

www.ciose.fr

Présentation C. Charreyre

- Associé au sein de CIO Systèmes Embarqués : Société d'ingénierie en systèmes embarqués - électronique et logiciel
- Responsable des technologies Linux embarqué
- Acteur Linux embarqué depuis 2000, OpenEmbedded puis Yocto depuis 2008
- Formateur Linux embarqué
- 30 ans dans l'embarqué et le monde Unix / Linux
- Fervent promoteur du logiciel libre
- Membre de Medinsoft – Commission Logiciel Libre

Contexte

Operating Systems et IoT

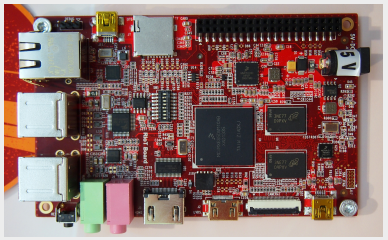
GNU/Linux, Windows

Cloud

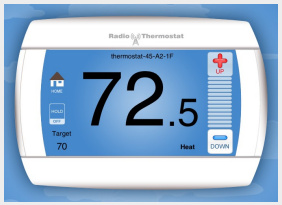
Embedded Linux

iOS Android

Gateways



OS dédiés : RIOT , Contiki, FreeRTOS, mbedOS ...



Devices

Quelques Middlewares IoT

THREAD par THREAD GROUP

IoTivity par OPEN INTERCONNECT CONSORTIUM SM

AllJoyn TM par ALLSEEN ALLIANCE

IoTivity et Allseen Alliance = projets collaboratifs de Linux Foundation

Yocto (Project) pour l'IoT

- Linux embarqué constitue l'un des OS impliqués dans les diverses couches de l'IoT
- Bâtir des solutions pérennes et robustes nécessite des outils professionnels
- Yocto = la référence actuelle dans la création de distributions Gnu/Linux embarquées
- Yocto Project fait également partie des projets collaboratifs Linux Foundation

Pourquoi un outil de build ?

Le challenge Linux embarqué

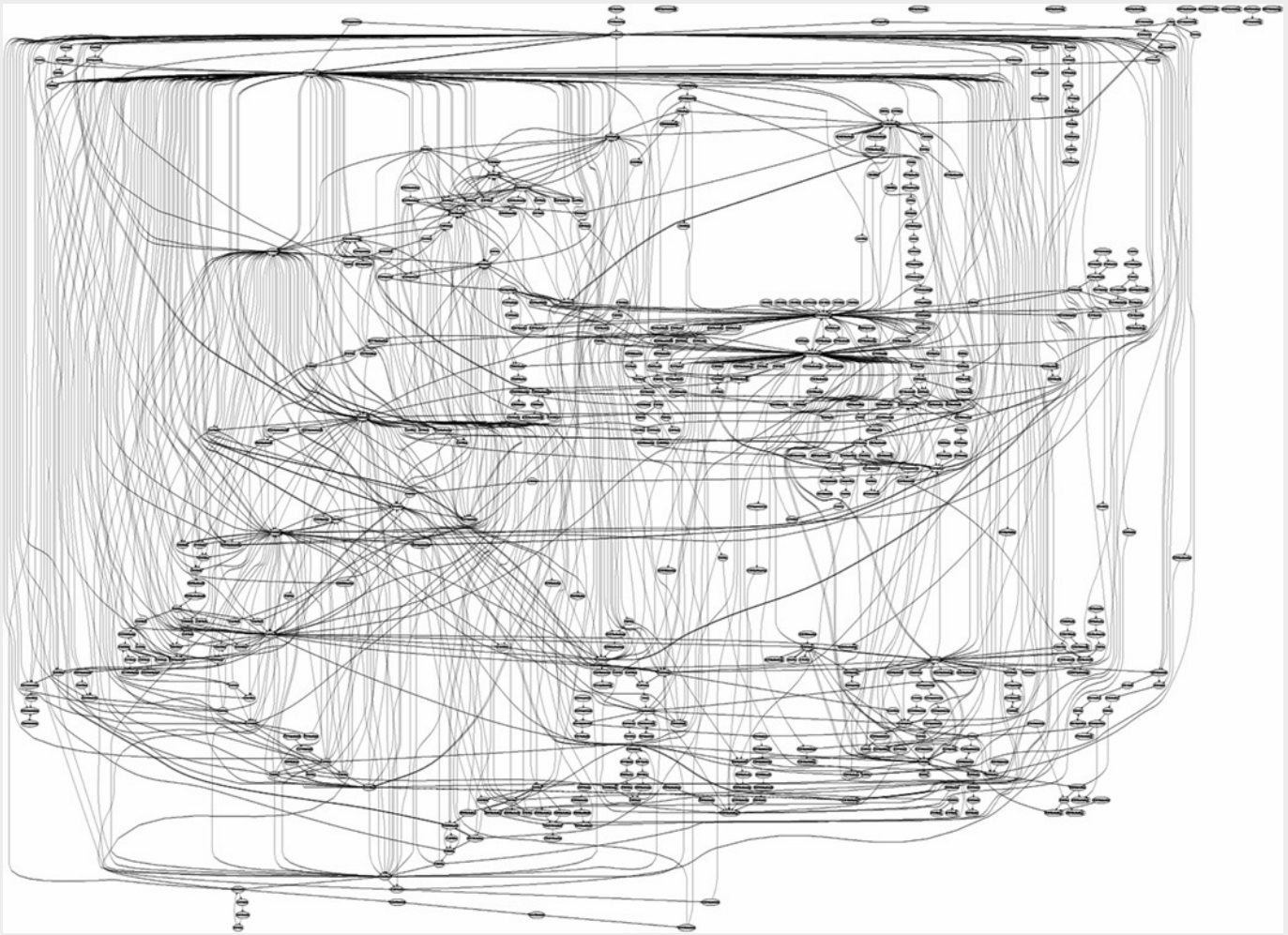
- Logique économique :
  - Réduire le Time To Market et optimiser les coûts
  - Utiliser largement des logiciels issus du riche écosystème Linux
  - Les assembler comme des briques modulaires, tels quels ou adaptés
- Le challenge = Gérer la complexité :
  - Un monde fragmenté aux multiples sources
  - Qu'il faut assembler en un ensemble cohérent : le device à réaliser
  - Mais chaque projet contributeur vit sa vie à son propre rythme : cycle de vie propre, sans gestion centralisée

Pourquoi un outil de build ?

- Pour éviter cela

Pourquoi un outil de build ?

- Pour gérer cela



Réseau de dépendances entre composants logiciels au cycle de vie autonome

Pourquoi un outil de build ?

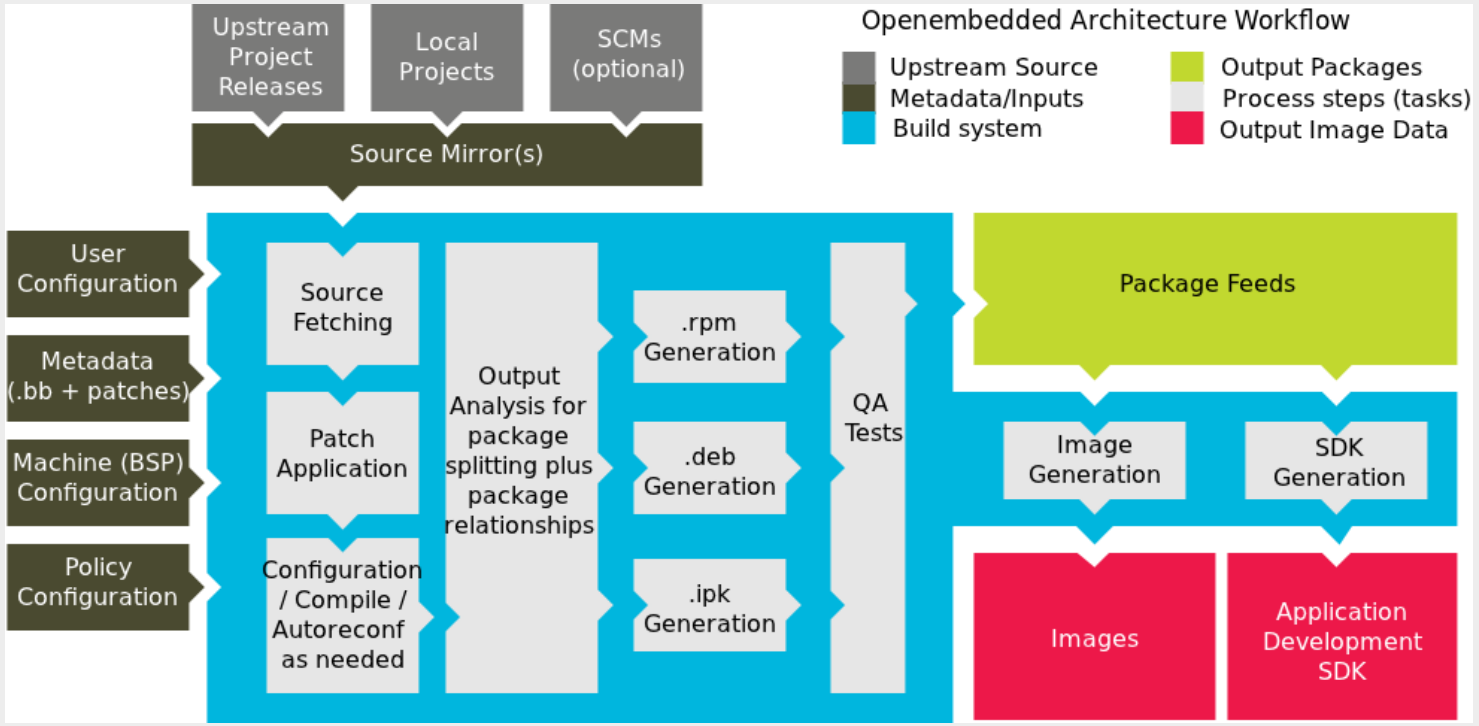
- Dépendances de nautilus (file manager) : 62 librairies

```
cch@cch-tosh: ~
Fichier  Édition  Affichage  Terminal  Aide

cch@cch-tosh:~$ ldd /usr/bin/nautilus
        linux-vdso.so.1 =>  (0x00007fff563a1000)
        libSM.so.6 => /usr/lib/libSM.so.6 (0x00007f75394b6000)
        libICE.so.6 => /usr/lib/libICE.so.6 (0x00007f753929b000)
        libXrender.so.1 => /usr/lib/libXrender.so.1 (0x00007f7539090000)
        libnautilus-extension.so.1 => /usr/lib/libnautilus-extension.so.1 (0x00007f7538e86000)
        libappindicator.so.0 => /usr/lib/libappindicator.so.0 (0x00007f7538c7d000)
        libgnome-desktop-2.so.17 => /usr/lib/libgnome-desktop-2.so.17 (0x00007f7538a52000)
        liblaunchpad-integration.so.1 => /usr/lib/liblaunchpad-integration.so.1 (0x00007f753884e000)
        libunique-1.0.so.0 => /usr/lib/libunique-1.0.so.0 (0x00007f7538641000)
        libdbus-glib-1.so.2 => /usr/lib/libdbus-glib-1.so.2 (0x00007f753841e000)
        libpthread.so.0 => /lib/libpthread.so.0 (0x00007f7538201000)
        libgailutil.so.18 => /usr/lib/libgailutil.so.18 (0x00007f7537ff9000)
        libgtk-x11-2.0.so.0 => /usr/lib/libgtk-x11-2.0.so.0 (0x00007f75379d6000)
        libgdk-x11-2.0.so.0 => /usr/lib/libgdk-x11-2.0.so.0 (0x00007f7537729000)
        libatk-1.0.so.0 => /usr/lib/libatk-1.0.so.0 (0x00007f7537508000)
        libgio-2.0.so.0 => /usr/lib/libgio-2.0.so.0 (0x00007f7537254000)
        libgdk_pixbuf-2.0.so.0 => /usr/lib/libgdk_pixbuf-2.0.so.0 (0x00007f7537038000)
        libcairo.so.2 => /usr/lib/libcairo.so.2 (0x00007f7536db5000)
        libpango-1.0.so.0 => /usr/lib/libpango-1.0.so.0 (0x00007f7536b6a000)
        libgobject-2.0.so.0 => /usr/lib/libgobject-2.0.so.0 (0x00007f7536922000)
        libgmodule-2.0.so.0 => /usr/lib/libgmodule-2.0.so.0 (0x00007f753671e000)
        libgthread-2.0.so.0 => /usr/lib/libgthread-2.0.so.0 (0x00007f7536518000)
        libgconf-2.so.4 => /usr/lib/libgconf-2.so.4 (0x00007f75362db000)
        libglib-2.0.so.0 => /lib/libglib-2.0.so.0 (0x00007f7535ffd000)
        libxml2.so.2 => /usr/lib/libxml2.so.2 (0x00007f7535cac000)
        libX11.so.6 => /usr/lib/libX11.so.6 (0x00007f7535976000)
        libexif.so.12 => /usr/lib/libexif.so.12 (0x00007f7535731000)
        libexempi.so.3 => /usr/lib/libexempi.so.3 (0x00007f7535444000)
        libselinux.so.1 => /lib/libselinux.so.1 (0x00007f7535226000)
        libm.so.6 => /lib/libm.so.6 (0x00007f7534fa3000)
        libc.so.6 => /lib/libc.so.6 (0x00007f7534c1f000)
        libuuid.so.1 => /lib/libuuid.so.1 (0x00007f7534a1a000)
        libindicator.so.0 => /usr/lib/libindicator.so.0 (0x00007f753480f000)
        libpangoft2-1.0.so.0 => /usr/lib/libpangoft2-1.0.so.0 (0x00007f75345e4000)
        libpangocairo-1.0.so.0 => /usr/lib/libpangocairo-1.0.so.0 (0x00007f75343d7000)
        libfreetype.so.6 => /usr/lib/libfreetype.so.6 (0x00007f7534151000)
        libfontconfig.so.1 => /usr/lib/libfontconfig.so.1 (0x00007f7533f1b000)
```

Yocto

Workflow de Yocto



Credit – Yocto Project

Workflow de Yocto

- Les entrées :
  - Code source des projets « upstream » :
    - Fetchés depuis le site projet ou depuis un miroir Yocto
    - Formats variés : archive, révision dans un SCM ...
    - Protocoles variés : http(s), ftp, svn, git …
  - Code source local (projet interne) :
    - Archive locale ou accès à un serveur SCM interne
  - Fichiers de configuration :
    - Caractéristiques de la machine cible (kernel, bootloader, format image, tuning compilateur ...)
    - Caractéristiques de la distribution (paquets inclus, versions, choix entre alternatives, choix libc …)
    - Caractéristiques des layers (détaillé ultérieurement)
    - Configuration du build : machine, distribution, layers actives, format paquet ...

Workflow de Yocto

- Le cœur :
  - Un moteur d'exécution de tâches écrit en Python : bitbake
  - Exécute automatiquement les tâches nécessaires à la fabrication de la cible fournie
  - Fonctionne en ligne de commande
  - Exemple : `bitbake core-image-minimal`

Workflow de Yocto

- Les sorties :
  - Les paquets binaires des composants utilisés (formats ipk, deb ou rpm)
    - Possibilité d'embarquer un gestionnaire de paquets sur la cible
    - Permet de travailler en différentiel pour mise à jour / enrichissement
  - L'image finale déployable sur la cible (formats variés : tar.bz2, ext3, ubi etc....)
  - Un SDK exportable pour les développeurs d'application
  - Un récapitulatif des paquets embarqués et de leur licences

Le moteur bitbake

- Un moteur écrit en Python : bitbake
- Un jeu de recettes pour fabriquer les paquets logiciels : fichiers texte
- Des dépendances entre paquets, décrites dans les recettes
- Pour chaque recette des tâches élémentaires
- Calcul de l'arbre des dépendances pour fabriquer les paquets dans le bon ordre

Principales tâches élémentaires

Tâche	Rôle
fetch	Téléchargement depuis dépôt upstream
unpack	Extraction dans répertoire travail
patch	Application de patches issus des recettes
configure	Configuration
compile	Compilation croisée
install	Installation dans tampon local au composant
populate_lic	Installation fichier(s) licence dans répertoire deploy
package	Fabrication des descriptifs des packages
populate_sysroot	Déploiement des paquets dans le sysroot
package_write_xxx	Création du paquet au format xxx

Anatomie d'une recette

- Une recette pour un ou plusieurs paquets (fractionnement pour optimisation de l'espace)
- Des variables d'environnement
- Des tâches élémentaires implicites ou explicites (pour modifier l'implicite) : langage shell et python
- Jeu de recettes géré par la communauté Yocto → solution au problème de complexité :
  - Cohérence entre versions de composants dépendants
  - Mise à jour des recettes fonction des évolutions upstream
  - Application de patches locaux quand cela est nécessaire

www.ciose.fr

Anatomie d'une recette (log4c)

```
SUMMARY = "a library of C for flexible logging to files,
syslog and other destinations"
SECTION = "libs"
HOMEPAGE = "http://log4c.sourceforge.net/"

LICENSE = "LGPLv2.1"
LIC_FILES_CHKSUM =
"file://COPYING;md5=7fbc338309ac38fefcd64b04bb903e34"

PR = "r1"

SRC_URI = "http://prdownloads.sourceforge.net/$
{PN}/log4c-${PV}.tar.gz \
file://oe.patch \
file://add-pkgconfig-data.patch \
"

inherit autotools pkgconfig binconfig

SRC_URI[md5sum] = "ca5412b7515d8901714ab7892323adb6"
SRC_URI[sha256sum] =
"6ed40a41307c26d052667e1661437394ab00e29cd24ff2640b502ba8
ab1e442b"
```

Anatomie d'une recette (ed)

```
DESCRIPTION = "a line-oriented text editor"
HOMEPAGE = "http://www.gnu.org/software/ed/"
BUGTRACKER = ""

LICENSE = "GPLv3+"
LIC_FILES_CHKSUM =
"file://COPYING;md5=f27defe1e96c2e1ecd4e0c9be8967949 \

file://ed.h;endline=20;md5=c708cda1b2e8d723d458690b7db03878 \

file://main.c;endline=24;md5=1bd039d59e04ee5f82adcc970144a2c3"

SECTION = "base"
PR = "r0"

# LSB states that ed should be in /bin/
bindir = "${base_bindir}"

SRC_URI = "${GNU_MIRROR}/ed/ed-${PV}.tar.gz \
           file://ed-1.2-build.patch"

SRC_URI[md5sum] = "9a78593decccaa889523aa4bb555ed4b"
SRC_URI[sha256sum] =
"211c67b0c4aae277d34b1c5f842db1952e468e5905142868e4718ac838f08a65"

do_configure() {
    ${S}/configure
}

do_install() {
    oe_runmake 'DESTDIR=${D}' install
}
```

Tâches explicites

www.ciose.fr

Anatomie d'une recette (image)

```
include recipes-sato/images/core-image-sato.bb
IMAGE_FEATURES += "debug-tweaks"
DISTRO_FEATURES += "pulseaudio"
WEB = "web-webkit"

# Add extra image features
EXTRA_IMAGE_FEATURES += " \
    ${SOC_EXTRA_IMAGE_FEATURES} \
    nfs-server \
    tools-debug \
    tools-profile \
    qt4-pkgs \
"

IMAGE_INSTALL += " \
    ${SOC_IMAGE_INSTALL} \
    cpufrequtils \
    nano \
    packagegroup-fsl-gstreamer \
    packagegroup-fsl-tools-testapps \
    packagegroup-fsl-tools-benchmark \
    packagegroup-qt-in-use-demos \
    qt4-plugin-phonon-backend-gstreamer \
    qt4-demos \
    qt4-examples \
    fsl-gui-extrafiles \
    "

export IMAGE_BASENAME = "fsl-image-gui"
```

Sélection de features

Sélection de composants logiciels

2015 SophiaConf
Le cycle azuréen de conférences Open Source

Bon à savoir

- Outil historiquement en mode console :
  - Mais apparition d'outils graphiques : Hob, Toaster
  - Plugin Eclipse : ADT
- Prévoir beaucoup de disque et de temps CPU :
  - Génération de la toolchain + libc par Yocto (temps CPU)
  - Conservation des étapes intermédiaires – optionnel mais utile – très gourmand en disque
- Connaissance de Python : non obligatoire mais un + pour comprendre / développer des recettes
- Connaissance des standards tels que autotools, pkgconfig etc... conseillée :
  - Plus du fait des logiciels gérés que de Yocto lui même

Conclusion

Conclusion

- Yocto fournit un framework complet pour développer des distributions embarquées riches et fiables :
  - Jeu de recettes large et forte communauté d'utilisateurs
  - Génération automatique d'image pré-paramétrées, prêtes à l'installation
  - Génération d'un SDK exportable aux développeurs
- Le framework et la communauté prennent en compte la complexité et la fragmentation de l'écosystème Linux
- Utilisé par des fondeurs majeurs (Intel, AMD, Freescale, Texas Instruments, Broadcom, Huawei …)
- Fondation de distribution Linux commerciales (Wind River Linux, Enea Linux, Mentor Graphics Embedded Linux ...)
- 2 release par an, sur une base prévisible (Printemps et Automne)
- Pour le domaine IoT, IoTivity compatible Yocto (meta-oic)

www.ciose.fr

Conclusion

- Yocto est un outil de packaging, la qualité globale dépend également de la qualité des projets upstream
  - Mais des corrections locales
  - Un effort de sélection de projets « sérieux »
- Ne pas négliger qu'il n'y a pas d'outil miracle :
  - Temps de prise en main initial
  - Courbe d'apprentissage pour passer par les stades :
    - J'utilise
    - Je comprends
    - Je modifie / je crée
- La documentation s'est professionnalisée (sur le site Web Yocto et distribuée avec l'outil).
- Possibilité de se faire accompagner par un spécialiste : réduction du Time To Market

Merci de votre attention

Questions ?