



C. Charreyre
christian.charreyre@cioinfoindus.fr

Open Embedded un framework libre pour des applications embarquées riches





Attribution-Noncommercial-Share Alike 2.0 France

● You are free:



to Share - to copy, distribute, display, and perform the work



to Remix - to make derivative works

● Under the following conditions:



Attribution. You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).



Non commercial. You may not use this work for commercial purposes.



Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.

● For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to

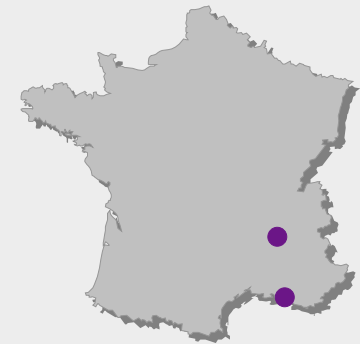
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/deed.en_US.

● Any of the above conditions can be waived if you get permission from the copyright holder.

● Nothing in this license impairs or restricts the author's moral rights.



- Société d'ingénierie en informatique industrielle et technique
- Au service de nos clients depuis 1990
- Une équipe de 15 spécialistes pour accompagner les projets industriels ou militaires
- La culture des systèmes ouverts et normalisés, l'expertise de l'embarqué et du temps réel
- Investissement sur Linux depuis 2000
Centre de Compétences créé fin 2001
- Siège à St Etienne, agence à Marseille
- Membre de Libertis, association de SSLL en région PACA

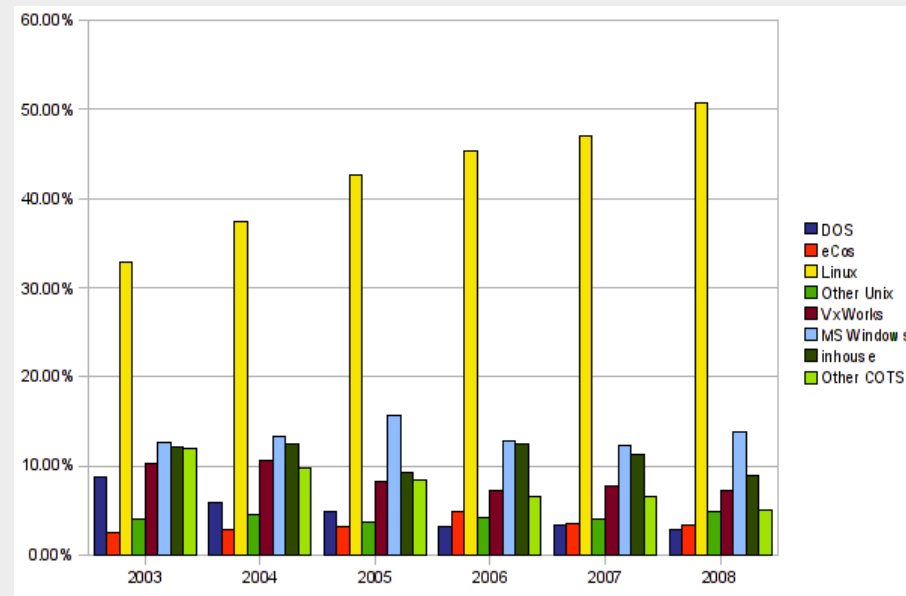


<http://www.libertis.org>

Linux sur le marché de l'embarqué



- Linux est un acteur en croissance rapide sur le marché de l'embarqué
- Il est utilisé dans plus de 50% des nouveaux design depuis 2008



Which OSes have been in used in your embedded designs during the past 2 years?

Source
Embedded Linux Market Survey 2009





- De nombreux appareils basés sur Linux :



Évolutions de Linux embarqué



- Il y a quelques années, Linux = moteur d'applications enfouies :
 - Besoin d'un kernel
 - Besoin glibc
 - Utilisation de busybox
 - Application embarquée home made sur ces bases
 - Peu de soucis de cohérence ou de complexité
- Besoins actuels : applications embarquées riches :
 - Moins de limitations de mémoire vive ou de masse
 - IHM riches : tactile, vidéo, image, culture « iPhone »
 - Nécessité d'assembler de manière cohérente de multiples briques logicielles de base



- Un monde fragmenté aux multiples sources
 - Bootloaders (UBoot, RedBoot, LILO, Grub, ...)
 - Kernel (kernel.org, fournisseur hardware, ...)
 - Bibliothèques de base (glibc ou alternatives réduites)
 - Bases applicatives (busybox, kits embarqués libres ou propriétaires,)
 - IHM (Qt, MicroWindows/NanoX, ...)
 - Multimédia (Mplayer, Gstreamer, Xine,)
 - Extensions temps réel (RTAI, Xenomai, ...)
- Qu'il faut assembler en un paquet cohérent :
votre application



- Faire attention au respect de licences multiples (GPL, LGPL, BSD, etc...)
 - Les connaître et les respecter
 - Adapter ce que l'on utilise à sa stratégie de publications de codes source
- Pour des applications riches, des Software Development Kits structurés sont conseillés :
 - Buildroot
 - Scratchbox
 - Open Embedded

Pourquoi OpenEmbedded ?



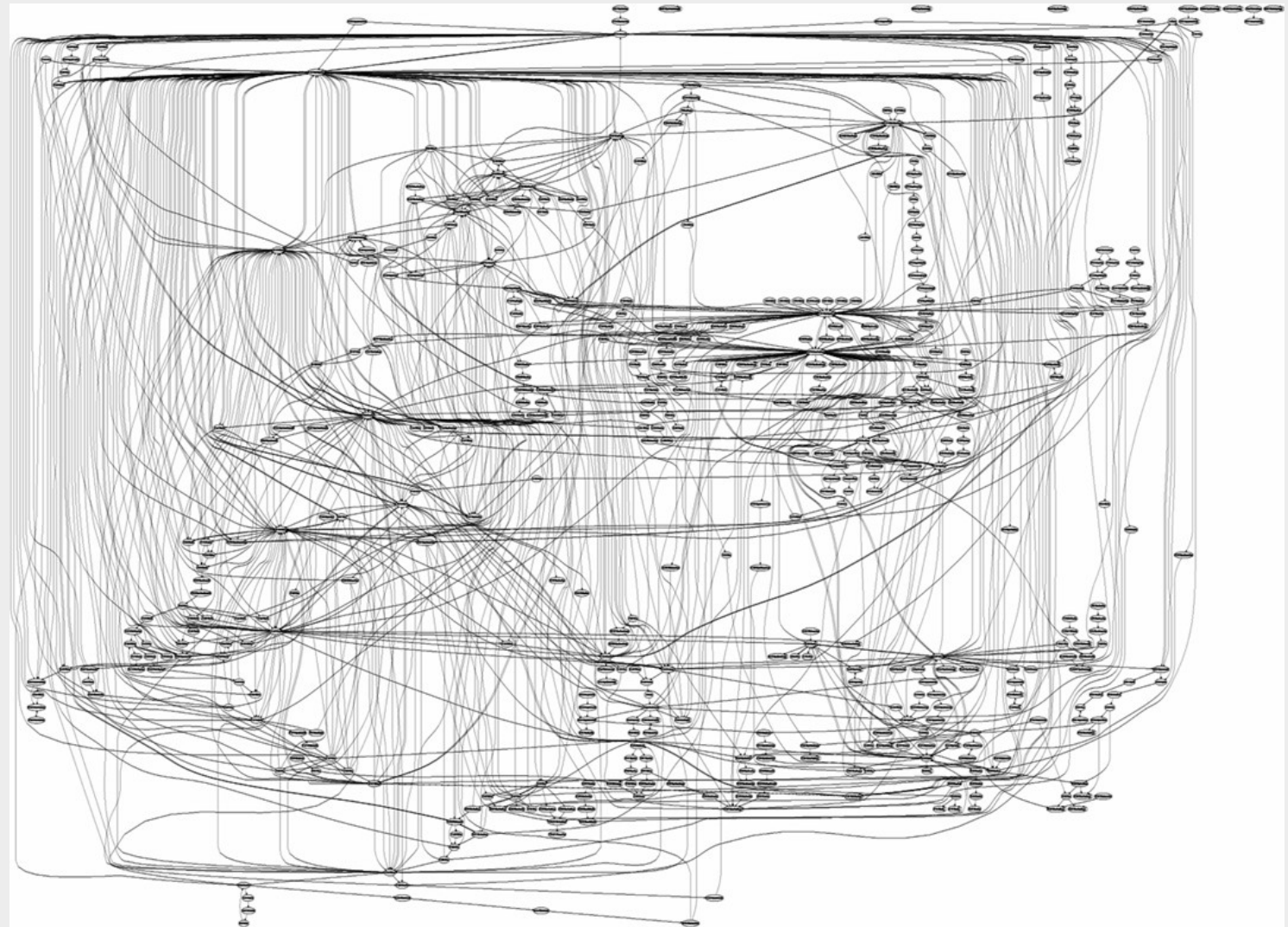
- Pour éviter cela



Pourquoi OpenEmbedded ?



- Pour gérer cela



Pourquoi OpenEmbedded ?



● Dépendances de nautilus : 72 bibliothèques

```
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)
```



Pourquoi OpenEmbedded ?



- Les « sorties » de l'outil :
 - La chaîne de compilation pour la cible (générée par OE) ▶▶
 - Le Software Development Kit qui se construit petit à petit : fichiers headers et bibliothèques partagées compilées pour la cible, outils natifs nécessaires à la compilation ▶▶
 - Des paquets logiciels binaires au format debian + un gestionnaire de paquets sur la cible (cohérences, dépendances) ▶▶
 - Une image finale prête à déployer (archive tgz, ubi, etc...) ▶▶
 - Une classification des logiciels par type de licence ▶▶

Les bénéfices d'OpenEmbedded



- Gestion intégrée des versions et de leurs dépendances
- Grande richesse logicielle : ▶▶
 - Firefox, Evolution
 - Gimp
 - Gnome, XFCE, OpenBox
 - VLC, Gstreamer, Totem
 - Samba, Apache
- Prise en compte de la cross compilation et des architectures non Intel (patches, configurations spécifiques)
- Prise en compte de l'internationalisation



Les bénéfices d'Open Embedded



- Indépendance vis à vis du poste de développement hôte :
 - Outils nécessaires à la compilation générés par OE puis utilisés (ex: python, dmake, pkg-config ...)
 - Pas d'utilisation des headers et bibliothèques de la distribution locale, même en architecture Intel
- Reproductibilité de la génération de l'image embarquée :
 - Remontée de tous les paramètres à la génération
 - Aucune intervention manuelle lors du déploiement sur la cible
- Communauté active





Comment fonctionne OpenEmbedded ?



- Un moteur écrit en Python : bitbake
- Un jeu de recettes pour fabriquer les paquets logiciels
- Une notion de classes pour mise en commun entre recettes
- Une notion de tâches = méta paquets
- Des dépendances entre paquets, décrites dans les recettes, ou déterminées automatiquement (bibliothèques partagées)
- Pour chaque recette des tâches élémentaires
- Calcul de l'arbre des dépendances pour fabriquer les paquets dans le bon ordre



Comment fonctionne OpenEmbedded ?



- Une notion de distribution :
 - Définit des versions préférentielles pour les paquets logiciels (cohérence de l'ensemble)
 - Définit des réglages spécifiques (fichiers de configuration)
- Prise en compte des spécificités de la plateforme :
 - Au niveau kernel : architecture, bootloader, modules kernel adaptés etc... ▶▶
 - Au niveau userland : paramétrages GCC, optimisations, floating point etc... ▶▶

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 : langage shell (basique) ou Python (+ évolué)



Anatomie d'une recette (at)



```
DESCRIPTION = "Delayed job execution and batch processing."
SECTION = "base"
LICENSE="BSD"
DEPENDS = "flex-native"
RCONFLICTS_${PN} = "atd"
RREPLACES_${PN} = "atd"

PR = "r3"

SRC_URI = "${DEBIAN_MIRROR}/main/a/at/at_${PV}-11.tar.gz \
          file://configure.patch \
          file://nonrootinstall.patch \
          file://use-ldflags.patch"

SRC_URI[md5sum] = "81dbae5162aaa8a398a81424d6631c77"
SRC_URI[sha256sum] =
"0d77c73a3c151a7da647dd924f32151e5ee4574530568fd65067882f79cd5a44"

inherit autotools

export LIBS = "-L${STAGING_LIBDIR}"

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





Anatomie d'une recette (cheese)



```
DESCRIPTION = "Take photos and videos with your webcam, with fun  
graphical effects"  
LICENSE = "GPLv2"  
  
DEPENDS = "gtk+ gstreamer gst-plugins-base libcanberra udev librsvg  
gnome-desktop eds-dbus"  
RRECOMMENDS_${PN} = "gst-plugin-gconfelements gst-plugins-good-meta  
gst-plugins-base-meta"  
  
inherit gnome  
  
SRC_URI[archive.md5sum] = "1599fded8a1797ea51fb010af4e6c45b"  
SRC_URI[archive.sha256sum] =  
"48f03470c6f527caa0e3b269d3afcff86ae0939a74f66ce030d4eed3bc3cbd9a"  
  
FILES_${PN} += "${datadir}/dbus-1"
```





Les principales tâches d'une recette



- Fetch : téléchargement des sources
- Unpack : extraction des sources
- Patch : application de patches additionnels fournis par la recette
- Configure : configuration
- Compile : compilation
- Stage : installation dans le SDK
- Install : installation dans un tampon local
- Package : création du (des) paquet(s) binaire





Anatomie d'une classe (qt4e)



```
DEPENDS_prepend = "${@["qt4-embedded ", ""][(bb.data.getVar('PN',  
d, 1) == 'qt4-embedded')]}"  
inherit qmake2  
  
QT_DIR_NAME = "qtopia"  
QT_LIBINFIX = "E"  
# override variables set by qmake-base to compile Qt/Embedded apps  
#  
export QMAKESPEC = "${STAGING_DATADIR}/${QT_DIR_NAME}/mkspecs/${  
TARGET_OS}-oe-g++"  
export OE_QMAKE_INCDIR_QT = "${STAGING_INCDIR}/${QT_DIR_NAME}"  
export OE_QMAKE_LIBDIR_QT = "${STAGING_LIBDIR}"  
export OE_QMAKE_LIBS_QT = "qt"  
export OE_QMAKE_LIBS_X11 = ""  
export OE_QMAKE_EXTRA_MODULES = "network"  
EXTRA_QMAKEVARS_PRE += " QT_LIBINFIX=${QT_LIBINFIX} "  
  
# Qt4 uses atomic instructions not supported in thumb mode  
ARM_INSTRUCTION_SET = "arm"
```





www.cioinfoindus.fr

Anatomie d'une tâche (task-xfce4-base)



```
DESCRIPTION = "All packages required for a base installation of  
XFCE 4.6.*"  
PR = "r1"
```

```
inherit task
```

```
RDEPENDS_${PN} = " \  
    xfwm4 \  
    xfwm4-theme-default \  
    xfce4-session \  
    xfconf \  
    xfdesktop \  
    xfce4-panel \  
    gtk-xfce-engine \  
    xfce-utils \  
    xfce4-panel-plugin-actions \  
    xfce4-panel-plugin-clock \  
    xfce4-panel-plugin-iconbox \  
    xfce4-panel-plugin-launcher \  
    xfce4-panel-plugin-pager \  
    xfce4-panel-plugin-separator \  
    xfce4-panel-plugin-showdesktop \  
    xfce4-panel-plugin-systray \  
    xfce4-panel-plugin-tasklist \  
    xfce4-panel-plugin-windowlist \  
    xfce4-settings \  
    xfce-terminal \  
    thunar \  
"
```





- 2 niveaux de versionning :
 - Version du logiciel (gérée par l'équipe projet qui développe ce logiciel)
 - Version de la recette OE (gérée par la communauté OE)
- Plusieurs recettes possibles pour un même logiciel (différentes versions du logiciel + svn/git) ▶▶
- Par défaut version la + élevée retenue – peut être contrôlé par paramétrage au niveau distribution
- Une seule version de recette disponible (la dernière) – les recettes OE sont gérées sous git



Gestion du versionning cible



- Gestionnaire de paquets sur la cible :
 - Installation
 - Suppression
 - Upgrade
- Gère les dépendances à l'installation / suppression
- Gère les versions logiciel + version recette :
 - Refus des downgrade sauf forçage





- Outil en mode console :
 - ne pas s'attendre à un IDE type Eclipse pour le moment
 - Améliorations futures au travers de Yocto project ???
- Prévoir beaucoup de disque et de temps CPU :
 - Génération de la toolchain + libc par OE (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 OE lui même



- 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
- Comme parfois (souvent ???) dans le monde du libre la documentation est en retrait / puissance de l'outil
- Comme toujours dans le libre on peut réinvestir en formation des équipes des économies sur les coûts récurrents : achats licence + maintenance
 - Investissement dans le capital humain des salariés vs achat de produits
- Se faire accompagner par un spécialiste : réduction du Time To Market

Quelques liens utiles



- Open Embedded :
<http://www.openembedded.org/>
- Distribution Angstrom :
<http://www.angstrom-distribution.org/>
- Distribution Poky Linux :
<http://www.pokylinux.org/>
- Yocto project (in progress) :
<http://yoctoproject.org/>

Pour aller plus loin



- Pour toute information complémentaire :
 - Visitez notre site Web <http://www.cioinfoindus.fr>
 - Contactez nous :
 - Tél : 04 95 05 19 41
 - Mail : <mailto:christian.charreyre@cioinfoindus.fr>
- Possibilité d'envoi des slides de la présentation sur demande



Démonstration



- Démonstration de la partie hôte (SDK)
- Gestion de paquets sur la cible

