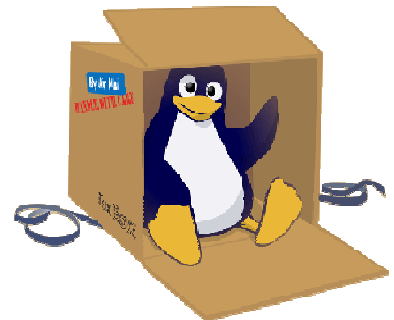


# Mettre en place un environnement de développement Openembedded Dreambox 7020



**Par Dominique DAMBRAIN, le 03/06/2005**

## 1.1 Objectifs :

Pour satisfaire votre curiosité, vous pouvez souhaiter, comme moi, mettre en oeuvre sur l'une de vos machines un environnement de développement pour Dreambox 7020.



## 1.2 Cet environnement a pour finalité :

- De mettre en place sur votre machine l'ensemble des outils et sources nécessaires à la production d'une 'image' de type **.nfi** exploitable en flash ou sur stick USB, pour une Dreambox 7020.
- Il est basé sur le projet 'OpenEmbedded', qui met à disposition sur le Web l'ensemble des composants nécessaires
- Les images **.nfi** produites sont installables sur votre Dreambox 7020
- Il est possible alors de personnaliser l'image, apporter sa touche personnelle, et disposer des mises à jour publiées sur le Net.

## 1.3 Pré-requis :

Il est probablement possible de mettre en place cet environnement sur de nombreux types de systèmes hôtes.

Toutefois, je l'ai pour ma part implanté sur un Linux RedHat FEDORA Core 3 (disponible gratuitement chez RedHat)

Rappelez-vous que la Dreambox utilise un système Linux, il semble donc plus naturel de développer sur un système Linux !

Toutefois, il sera nécessaire de construire un système de Cross Compilateur, car le processeur cible de la Dreambox est un PowerPC de IBM, alors que l'environnement de compilation sera installé sur une machine à base de Intel Pentium, dans mon cas !  
Mais ne vous inquiétez pas : l'environnement OpenEmbedded, lorsqu'il est correctement paramétré, se charge de tout !

Cet OS Linux FEDORA Core 3 présente l'avantage d'inclure la grande majorité des outils nécessaires au développement, dans le cadre du projet OpenEmbedded, dans la mesure où l'on installe la version "Workstation".

Vous aurez toutefois à installer en plus, à partir des CDs d'installation FEDORA, le produit optionnel "Subversion", appelé aussi 'svn'

Il est fortement recommandé de disposer sur la machine hôte de, au minimum :

- 512 Mo de mémoire RAM ( 640 Mo recommandé, pour éviter tout 'swap')
- 30 Gb de disque disponible, après installation du système.

Le téléchargement des sources OpenEmbedded depuis le Web nécessite la mise en place du produit 'Bitkeeper', ou de sa version libre 'Bitmover'. Vous pouvez télécharger la version installable sur Fedora Core 3 à l'URL suivante :

<http://www.dambrain.homelinux.net/Dreambox/Openembedded/bk-3.2.4-x86-glibc23-linux.bin>

Pour installer Bitkeeper : Rendez exécutable le fichier téléchargé et exécutez-le en étant root :

```
chmod 755 ./bk-3.2.4-x86-glibc23-linux.bin
su -
./bk-3.2.4-x86-glibc23-linux.bin
exit          (ne restez pas loggé root : ce serait dangereux pour la suite des opérations !)
```

Enfin, pour la mise en œuvre Openembedded, il est recommandé d'installer l'addon '**Compilateur Python Just-in-Time Psyco**', ce qui améliore grandement les performances de la phase de parsing des fichiers de type .oe (cette phase est appelée à chaque invocation d'une commande 'oemake'). Vous pouvez télécharger la version installable sur FEDORA Core 3 à l'URL suivante :

<http://www.dambrain.homelinux.net/Dreambox/Openembedded/psyco.tar>

Pour installer Psyco : Décompressez l'archive dans un répertoire de votre choix puis allez dans ce répertoire :

```
cd ./psyco-1.3
su -          ( la mise en place doit se faire avec les droits de root)
python setup.py install (python doit bien sûr avoir été préalablement installé, et être accessible !)
exit        (ne restez pas loggé root : ce serait dangereux pour la suite des opérations !)
```

## 1.4 Mise en place de l'environnement OpenEmbedded :

La mise en place est relativement simple, à condition de respecter certaines règles :

- Créer un nouvel utilisateur, au sens Unix : Dans mon exemple, j'ai créé un utilisateur 'stuff', dont la Home directory est /stuff (en fait, un filesystem dédié, sur un disque dédié !)
- Toujours effectuer les opérations relatives à cet environnement en étant identifié sous cet utilisateur ( **JAMAIS EN TANT QUE ROOT** )

En effet, grâce à l'ensemble des manipulations, il va être créé dans cet environnement un 'Cross Compilateur' capable de générer des binaires exécutables à destination de la Dreambox.

Or, certaines ressources, telles que le compilateur GNU C (gcc) ne doivent pas être ambiguës, faute de quoi cela pourrait avoir pour conséquence de détruire l'environnement technique de la machine hôte (FEDORA Core 3 sur Intel x86).

N'étant pas identifié 'root', une erreur de configuration ou setting ne risque pas de mettre en danger vos répertoires 'système hôte' !

---

Dans le répertoire Home de cet utilisateur, vous devrez installer les scripts suivants :

Les scripts de mise en place, décrits ci-dessous, sont téléchargeables à l'URL suivante :

<http://www.dambrain.homelinux.net/Dreambox/Openembedded/scripts.tar>

Pour les installer, téléchargez le fichier 'scripts.tar' dans la Home de l'utilisateur

Loggez-vous avec cet utilisateur (pas ROOT !)

entrez la commande : **tar xvf scripts.tar** (il va créer les répertoires **./bin** et **./build** s'ils n'existaient pas)

Changez les propriétés de ces scripts pour qu'ils soient exécutables :

```
chmod 744 ./env.sh \  
./install.sh \  
build/env.source \  
bin/go \  
bin/comp_enigma
```

- 1- \$HOME/env.sh (setting des variables d'environnement requises)
- 2- \$HOME/local.conf (template de configuration des variables Openembedded)
- 3- \$HOME/install.sh (Mise en place initiale de Openembedded)
- 4- \$HOME/build/env.source (Configuration pur DM7020)
- 5- \$HOME/bin/go (exemple personnel de lancement d'une compilation)
- 6- \$HOME/bin/comp\_enigma (exemple personnel de re-compilation forcée pour enigma)

Normalement, ces scripts ne requièrent aucune modification (tout y est relatif à \$HOME de l'utilisateur).

Exécuter alors, dans le shell courant, la commande :

**`./env.sh`**

Ceci positionne et exporte les variables nécessaires, dans le shell courant (raison du 1er .)

Elles seront conservées pendant tout le temps de session de ce shell !

Ensuite, lancer le script install.sh :

**`./install.sh`**

Ce script va créer les sous-répertoires nécessaires, et utiliser Bitkeeper pour télécharger les sources et snapshots nécessaires à la création du contexte de développement. Selon les performances de votre accès Internet, cela peut nécessiter plusieurs minutes.

Ce script install.sh n'a normalement pas à être lancé plusieurs fois : Il assure la mise en place initiale. D'ailleurs, des lancements successifs provoquent des erreurs d'exécution, car la création des répertoires rencontre des problèmes (ils sont déjà créés !)

## 1.5 Génération d'une image Dreambox 7020 :

Pour faciliter le lancement de cette opération, j'ai mis à votre disposition un script **`/bin/go`**

Il suffit donc maintenant de taper **`go`**

Il va exécuter les settings nécessaires, puis aller dans le sous répertoire 'build', et lancer 'oemake' (Openembedded make)

Les différentes phases de cette génération :

1- Installation et compilation d'un environnement de développement relatif au système hôte ( Linux FEDORA Core 3 sur Intel x86)

Eh oui : il recrée son **PROPRE** environnement de compilation, et n'altère donc en rien celui de la machine hôte !

Ceci évite les problèmes de compatibilité de versions d'outils entr'eux ( Kernel version X, GLIBC version Y, etc....)

2- Grâce aux outils installés durant l'étape 1, mise en place d'un 'cross compilateur' pour processeur PowerPC Dreambox

**Grâce à son PROPRE environnement de compilation, il installe un compilateur pour POWERPc !  
Ceci évite au maximum les problèmes de compatibilité !**

3- Compilation des composantes spécifiques à la gestion d'une Dreambox ( enigma, jeux, plugins, etc...)

4- Génération de l'arborescence de répertoires nécessaire à une Dreambox en mémoire flash

5- Création d'une 'image' (type .nfi) installable sur une Dreambox 7020.

**NOTE :**

Les étapes 1 et 2 peuvent nécessiter, suivant la puissance du processeur hôte, les performances de votre disque et surtout la RAM disponible, entre au moins **2 heures et jusqu'à 5 ou 6 heures !**

L'étape 3, de la même manière, peut nécessiter **1 ou 2 heures**

Les étapes 4 et 5 ne réclament en général que quelques minutes.

Si vous modifiez des sources , développez un plugin, ou apportez toute modification, seules certaines étapes seront nécessaires à nouveau :

- Quelques étapes de 3 (sauf si vous modifiez le noyau Linux lui-même !), ainsi que 4 et 5

Les phases 1 et 2 ne seront plus requises, sauf si vous créez un nouvel environnement, dans un répertoire différent .

Alors, patience et longueur de temps font plus que force ni que rage !

## 1.6 Problèmes rencontrés durant la mise en œuvre :

Les fichiers composant Openembedded contiennent quelques données 'périmées', pointant sur des sites n'existant plus, ou ne proposant plus les versions requises par l'environnement Dreambox.

Ainsi, le problème essentiel à résoudre, lors de mes premières tentatives de mise en place, a été de "mettre la main" sur les packages en question, à la bonne version !

J'en profite pour dire que si j'ai pu aboutir, c'est grâce à l'aide précieuse que m'a apportée l'équipe des développeurs 'Power of Dream' (image P.O.D.) , en me sortant de chaque impasse au prix de beaucoup de patience et de dévouement !

Pour éviter qu'ils n'aient à réitérer la même opération avec de nombreux candidats , j'ai décidé de proposer en téléchargement sur mon site L'ENSEMBLE DES ARCHIVES nécessaires à cette mise en place :

Recommandation : Merci de ne pas télécharger de manière systématique tous les packages, mais seulement ceux qui s'avèrent inaccessibles lors de votre compilation. En effet, la bande passante n'est pas infinie, et les serveurs Openembedded sont mieux équipés que moi, à ce sujet !

Mais au moins, vous n'aurez pas à chercher longuement ceux qui vous manquent : Tous ceux qui ont contribué au succès sur ma machine y sont !

### URL de téléchargement des snapshots :

[http://www.dambrain.homelinux.net/Dreambox/Openembedded/snapshots\\_20050524](http://www.dambrain.homelinux.net/Dreambox/Openembedded/snapshots_20050524)

Je précise qu'il s'agit d'un snapshot qui a prouvé sa cohérence **à la date du 24 Mai 2005.**

Je n'assume aucune responsabilité si vous travaillez sur la base d'une version CVS ou Openembedded ultérieure : Il vous appartiendra alors d'identifier les éventuels 'manquants', et les télécharger ailleurs !

Chaque package ( fichier .tar.gz ou .tar.bz2) doit être installé dans le répertoire \$HOME/sources

En relançant la commande 'go', il va re-tenter le téléchargement depuis OE ou CVS, mais ayant les mêmes paramètres qu'auparavant, cela va bien sûr échouer... Par contre, il va détecter qu'une version adéquate du package est déjà en place, et va finalement en extraire les fichiers, pour continuer le 'oemake' !

### 1.7 Forcer la recompilation d'un package :

Le système de make proposé par Openembedded tient à jour une liste des opérations effectuées, dans un répertoire :

\$HOME/build/tmp/stamps

Pour forcer la recompilation individuelle d'un composant, d'un package ou d'un plugin, il suffit d'effacer le stamp correspondant, et relancer oemake (script 'go' )

A titre d'exemple, je vous propose le script [\\$HOME/bin/comp\\_enigma](#) (téléchargé avec le fichier 'scripts.tar' ci-dessus)

Plutôt que de faire les opérations nécessaires manuellement, en activant ce script [comp\\_enigma](#), il va :

- supprimer les fichiers 'stamp' nécessaires
- effectuer un 'touch' sur les sources de enigma
- et lancer le script 'go' -> compilations, packaging, et ré-génération de l'image .nfi

### 1.8 Forcer la mise à jour Openembedded, CVS Dreambox :

en cours de développement/tests : sera renseigné plus tard

### 1.9 Modifier la liste des composants à compiler lors d'une commande 'oemake' (personnalisation )

en cours de développement/tests : sera renseigné plus tard

### 1.10 Modifier la liste des composants et plugins à inclure dans une image .nfi

en cours de développement/tests : sera renseigné plus tard

### 1.11 Ajouter son propre plugin aux packages et à l'image générée :

en cours de développement/tests : sera renseigné plus tard