



RPM (RedHat Package Manager), URPM (User RPM) et apt-get (User RPM)

par Jean-Christophe 'Jice' Cardot

révision par Jean-Marc Lichtle, puis Jice et Anne

Installer des programmes sous Linux est très simple quand on utilise les bons outils...

Ce document ne vise pas à remplacer la page de man de RPM, mais plutôt de donner une courte référence des commandes les plus utiles pour gérer vos paquetages à ce format.

Introduction

Le format RPM (RedHat Package Manager), a été, comme son nom l'indique, créé à l'origine pour la distribution RedHat. Depuis, de nombreuses distributions l'ont utilisé, on peut citer par exemple Mandrake ou SuSe. D'autres formats de packages dans le monde Linux sont DEB (pour Debian) et TGZ (pour Slackware).

Qu'est-ce qu'un package ? C'est un fichier (une archive, un peu comme un fichier .tar ou .zip) qui contient tous les fichiers appartenant à une application, une bibliothèque, etc. ainsi que des scripts de contrôle qui sont exécutés lors de l'installation ou de la désinstallation de l'application. Ce fichier contient également toutes les dépendances à d'autres applications, bibliothèques, etc.

Note : on parle indifféremment de package, paquetage ou même paquet pour désigner le fichier qui contient l'ensemble des fichiers d'un programme (le fichier rpm lui-même)

Il existe de nombreux utilitaires graphiques tels que Kpackage, GnoRPM, Midnight Commander, etc. qui permettent d'installer et de gérer les packages RPM. L'installation depuis la ligne de commande reste toutefois la solution la plus rapide et la plus efficace pour ce travail.

La plus grande partie des commandes décrites peuvent être exécutée par un utilisateur lambda. Toutefois les commandes qui installent, mettent à jour ou suppriment des paquetages nécessitent des droits d'administrateur (root).

Nous allons voir ici deux méthodes pour gérer les paquets RPM :

- [le programme rpm](#),
- [les programmes urpm*](#) de Mandrake.

Les noms de paquetages sont très longs et donc très pénibles à orthographier correctement. Linux offre toutefois des aides à la frappe :

- lorsque vous installez un RPM, ou travaillez sur un RPM non encore installé, utilisez la complétion de ligne de commande en tapant les premières lettres du nom du fichier RPM puis la touche TAB. Si la correspondance reste équivoque, Linux vous donnera les possibilités trouvées, retapera votre ligne et vous laissera ajouter quelques lettres pour lever l'indétermination.
- lorsque vous travaillez sur un RPM qui est déjà installé, la commande `rpm` s'adresse à la base de données des paquetages installés, qui consigne les noms courts en plus des noms de fichiers complets. L'indication du numéro de version n'est donc en général pas nécessaire (sauf si le même package est installé 2 fois avec 2 versions différentes).

Les commandes rpm usuelles

Pour installer un nouveau paquetage :

```
[jice@taz jice]$ rpm -ivh toto.rpm
```

Attention, si vous installez un paquetage par cette méthode et qu'il existe déjà sur votre système dans une version inférieure, vous risquez d'avoir des problèmes pour le désinstaller (voir plus bas). Par contre, pour installer une bibliothèque simultanément en deux version différentes, utilisez cette syntaxe.

Les options `h` et `v` ne sont pas obligatoires, `-i` est suffisant pour faire une installation. L'option `h` provoque l'affichage d'une barre de progression (ça fait très pro sur l'écran !) et `v` rend l'installation bavarde, ce qui fournit des messages plus explicites en cas d'erreur. L'option `-vh` peut être appliquée sur toutes les commandes rpm ci-après afin de rendre rpm plus bavard.

Pour mettre à jour un paquetage :

```
[jice@taz jice]$ rpm -Uvh toto.rpm
```

Avec un "U" comme "Upgrade" (mise à jour en anglais).

Attention, ceci ne s'applique pas pour upgrader un noyau, voir la [rubrique noyau](#) à ce sujet.

Enfin, pour remplacer un paquetage par un paquetage plus ancien ("downgrader"), la commande à utiliser est :

```
[jice@taz jice]$ rpm -Uvh --oldpackage toto.rpm
```

Pour supprimer un paquetage :

```
[jice@taz jice]$ rpm -e toto  
[jice@taz jice]$ rpm -e toto-version
```

Pour faire des requêtes sur les paquetages :



Le "e" correspond à "erase" (effacer en anglais).

La syntaxe complète (avec numéro de version) permet de distinguer deux versions d'un même paquetage qui auraient été installées ensemble.

Pour faire des requêtes sur les paquetages :

Afficher la liste de tous les paquetages déjà installés J'imagine que "a" doit signifier "all" ;-) :

```
[jice@taz jice]$ rpm -qa
```

et

```
[jice@taz jice]$ rpm -qa | less
```

Pour afficher la liste page par page...

Trouve un paquetage identifié par son nom (toto) :

```
[jice@taz jice]$ rpm -qa | grep toto
```

La recherche peut s'appliquer à une partie seulement du nom, par exemple `rpm -qa | grep 'util'` vous donne la liste de tous les RPM installés tels que `util-linux`, `nautilus`, `bind_utils` etc.. Attention la syntaxe est sensible à la distinction majuscules / minuscules !

Pour trier par date d'installation :

```
[jice@taz jice]$ rpm -qa --last
```

Pour trier par taille :

```
[jice@taz jice]$ rpm -qa --queryformat '%{name} %{size}\n' | sort -n +1 | column -t
```

Jolie ligne de commande n'est-ce pas ? ;-)

Obtenir la liste des fichiers contenus dans un paquetage toto qui est déjà installé :

```
[jice@taz jice]$ rpm -ql toto
```

On dira que "l" est l'initiale de "list".

Dans la foulée de la syntaxe précédente, celle-ci donne les informations relatives à ce paquetage, une brève description de ce que fait le paquetage, des dépendances qui doivent être satisfaites, etc. :

```
[jice@taz jice]$ rpm -qi toto
```

On parie que "i" est mis pour "information" ?

La même chose sur un paquetage pas encore installé :

```
[jice@taz jice]$ rpm -qip toto.rpm
```

Avec un "p" comme "package".

Nota :

- L'option `p` s'applique aussi à d'autres syntaxes, par exemple `rpm -qlp toto.rpm` pour avoir la liste des fichiers d'un paquetage non encore installé.
- Les options de requêtes peuvent se cumuler. Par exemple `rpm -qlip toto.rpm` donnera à la fois les infos et la liste des fichiers du paquetage `toto.rpm`.

Retrouver le paquetage d'origine d'un fichier :

```
[jice@taz jice]$ rpm -qf /usr/lib/toto.txt
```

Il peut se produire que vous vous trouviez en face d'un fichier dont vous vous demandez quel est le paquetage qui a bien pu installer ce fichier. Pas de panique ! Là aussi la commande RPM peut vous être d'une aide précieuse. Ici le "f" signifie fichier (file en anglais).

Cette commande fonctionne sur les packages installés. Oui, mais si on veut trouver un fichier parmi les paquetages non installés ?

Si vous utilisez Mandrake (voir [la partie sur urpm plus bas](#)), la commande suivante fera l'affaire :

```
$ urpmf le_fichier
```

Sinon, placez-vous dans le répertoire contenant les `.rpm` et faites :

```
$ for i in *.rpm ; do rpm -qpli $i | grep file && echo $i ; done
```

Tester l'installation d'une paquetage :

```
[jice@taz jice]$ rpm -i --test toto
```



Effectue simplement une vérification sans installer le paquetage toto. Permet essentiellement de vérifier si l'installation pourrait se dérouler sans encombre ou alors si des dépendances non satisfaites risquent de faire avorter l'installation.

Si rpm ne veut pas installer le paquetage toto

Il existe plusieurs options. Premièrement : ne pas vérifier les dépendances.

```
[jice@taz jice]$ rpm -i --nodeps toto
```

Deuxièmement : forcer l'installation, en cas de conflit avec certains autres rpm.

```
[jice@taz jice]$ rpm -i --force toto
```

Enfin, combinez les deux options pour vraiment obliger rpm à procéder à l'installation.

Si rpm ne veut pas désinstaller le paquetage toto

Il suffit alors de faire :

```
[jice@taz jice]$ rpm -qa | grep toto
```

Affiche la liste des paquetages dont le nom contient toto, par exemple :

```
toto-1.1  
toto-1.2
```

```
[jice@taz jice]$ rpm -e toto-1.1
```

Désinstalle le paquetage désigné par son nom **et** son numéro de version.

Extraire des fichiers d'un RPM :

Certains outils comme Midnight Commander (mc, ou gmc) permettent d'ouvrir des fichiers rpm, de se déplacer dedans et de copier des fichiers vers d'autres répertoires.

Vous pouvez aussi convertir le rpm en archive cpio (l'ancêtre de tar), par la commande `rpm2cpio`, et ensuite utiliser la commande `cpio` pour extraire le ou les fichiers.

Installer un RPM dans un autre système Linux que celui qui tourne :

Vous avez plusieurs systèmes Linux sur votre machine, ou bien vous avez bouté avec le CD ou une disquette de sauvegarde. Vous souhaitez installer un rpm dans un autre système dont la partition racine est montée dans /mnt/racine. La commande à taper est :

```
$ rpm -i --root=/mnt/racine /chemin/package.rpm
```

Vous pouvez utiliser l'option `--root=/mnt/racine` dans toutes les commandes rpm, pour faire des recherches, etc. avec rpm sur l'autre système Linux.

ET TOUJOURS :

```
[jice@taz jice]$ man rpm
```

L'accès à la page du manuel, le réflexe qu'on devrait toujours avoir...

Quelques précisions concernant rpm

Quelle est la différence entre SRPM et RPM ?

Les paquetages SRPM (Source RPM) contiennent les sources d'un logiciel prêtes à être recompilées et transformées en paquetage rpm. Afin de reconstruire un paquetage rpm à partir d'un srpm (fichier `.src.rpm`), il faut utiliser la commande :

```
# rpmbuild --rebuild paquetage.src.rpm
```

Le paquetage rpm résultant se trouvera dans `/usr/src/RPM/RPMS/<arch>` où `<arch>` est `i386`, `i586`, `ppc`, etc. suivant votre architecture.

Pour simplement installer le paquetage, sans reconstruire le fichier `.rpm`, faire :

```
# rpmbuild --recompile paquetage.src.rpm
```

Cela peut être très utile pour installer un programme qui a été compilé avec des versions de bibliothèques différentes des vôtres et qui refuse donc de s'installer, tout en conservant l'intégrité de votre base de données des rpm (tout programme installé devrait être dans la base, mais si vous installez à partir des sources `.tar.gz` par `./configure && make install`, cela ne sera pas le cas.) Par exemple, vous pouvez récupérer un srpm de Mandrake Cooker (la version de développement) et essayer de le reconstruire sur une Mandrake 8.2.

Que sont les fichiers *.rpmsave et *.rpmnew ?



Note : ce processus nécessite que les paquetages `-devel` nécessaires aient été installés, ainsi que les compilateurs utilisés, etc. Voir l'article sur la [compilation](#).

Que sont les fichiers *.rpmsave et *.rpmnew ?

Lorsque rpm installe un paquetage, il peut soit conserver les anciens fichiers de configuration ; les nouveaux seront alors renommés en *.rpmnew. Si rpm remplace les anciens fichiers de configuration par des nouveaux, alors ce sont les anciens qui seront renommés en *.rpmsave.

Quelle est la différence entre "installer" et "mettre à jour" ?

La mise à jour (*upgrade* en anglais) remplace l'ancien paquetage par le nouveau, tandis que l'installation conserve si possible l'ancienne version du programme et installe la nouvelle en parallèle.

Cela peut être fort utile pour les bibliothèques : lorsque vous essayez de mettre à jour une bibliothèque, vous pouvez rompre des dépendances avec les programmes déjà installés (s'il y a un gros saut de version). Afin de ne pas avoir de problèmes, au lieu de mettre à jour la nouvelle version, installez-la ; ainsi l'ancienne version restera présente et les anciens programmes tourneront sans problème. Cela ne pose aucun problème d'avoir plusieurs versions d'une bibliothèque installée sur un système.

Que sont les paquetages `-devel` ?

Vous avez sans doute remarqué que souvent, pour un package `toto.rpm`, vous aviez un deuxième package `toto-devel.rpm`.

`toto.rpm` contient le logiciel, bibliothèque, etc. lui-même, c'est à dire la version que vous allez utiliser tous les jours.

`toto-devel.rpm` contient des fichiers (les "entêtes", etc.) qui permettent de compiler des programmes qui utilisent `toto`. Ainsi, toutes les bibliothèques ont leur paquetage `-devel`.

Vous avez besoin d'installer les paquetages `-devel` uniquement si vous désirez compiler des logiciels, que ce soit d'après l'archive `tar.gz` ou le paquetage `src.rpm`.

Les commandes `urpm*` de Mandrake

Le programme rpm de RedHat souffre de nombreuses limitations que Mandrake a dépassées, en créant les outils `urpm*` (User RPM – à partir de Mandrake 7.0) :

- résolution automatique et installation/désinstallation automatique des paquetages dépendants (rpm indique seulement le nom de paquetages manquants)
- `urpm` connaît l'ensemble des paquetages installables depuis différentes sources (les CD, serveurs ftp de mise à jour, [Penguin Liberation Front](#) ...)
- `urpm` permet d'installer des paquetages depuis internet lorsque ceux-ci sont plus récents que sur les CD (depuis Mandrake 8.0)
- `urpm` permet d'installer des paquetages en donnant seulement un nom incomplet (exemple : `urpmi koffice` pour chercher et installer Koffice, `urpmi mplayer`, etc.)
- etc.

Sur chacune des commandes dont on va parler ci-dessous, je vous conseille de lire la page de man (`man urpmi`, `man urpmf`, etc.) afin d'en apprendre plus.

Toutes les actions décrites ici peuvent également être réalisées de manière graphique, avec le Gestionnaire de Programmes (Software Manager), qui est une façade ou frontend aux programmes `urpm*`. Je vous conseille de tester les deux manières de faire, car contrairement aux apparences, la ligne de commande peut être plus simple et/ou rapide.

Configurer `urpm`

Ajouter / mettre à jour une source de paquetages

Après avoir installé votre distribution, les CD ont été créés dans la base de données de `urpm` en tant que source de paquetages. Cependant, vous pouvez ajouter autant de sources que vous le désirez. Supposons par exemple que vous ayez un répertoire "incoming/rpms" dans lequel vous placez tous les rpm que vous récupérez sur internet. Vous pouvez l'ajouter à votre base de données `urpm` par la commande :

```
# urpmi.addmedia mes_rpm file:///home/jice/incoming/rpms
```

Où `mes_rpm` est le nom (arbitraire) que vous donnez à votre source de paquetages.

Par la suite, vous pourrez mettre à jour cette source par :

```
# urpmi.update mes_rpm
```

De même, pour ajouter une source ftp de paquetages, on donne l'URL du fichier `hdlist.cz` sur le serveur. Exemple pour le [Penguin Liberation Front](#) et une Mandrake 9.0 :

```
# urpmi.addmedia plf ftp://ftp.easynet.fr/plf/9.0 with hdlist.cz
```



Et de la même façon, pour mettre à jour la source de temps en temps :

```
# urpmi.update plf
```

Note : le fichier de configuration de urpm est `/etc/urpmi/urpmi.cfg` et les bases de données dans `/var/lib/urpmi`.

Supprimer une source de paquetages

Pour supprimer la source mes_rpm, taper :

```
# urpmi.removemedias mes_rpm
```

Installer avec urpmi

Installer depuis les sources urpm

Pour installer un logiciel avec urpm, rien de plus simple. Vous tapez simplement la commande **urpmi** suivie d'une partie du nom du logiciel. Exemple :

```
# urpmi mplayer
```

urpmi va alors chercher la version la plus récente du package correspondant, regarder s'il doit installer des packages dépendants (et vous en demander confirmation), et installer l'ensemble des packages.

Dans le cas où l'argument donné à urpmi est ambigu, urpmi renvoie le nom des packages correspondants ; il ne vous reste plus qu'à choisir dans la liste et relancer urpmi avec le bon nom de package.

Si vous voulez installer le package qui fournit une librairie (par exemple), utilisez le modificateur `-p` :

```
# urpmi -p libe2p.so.2
```

urpmi va alors rechercher quel paquetage fournit `libe2p.so.2` et l'installer (ici `libext2fs2`).

Installer directement un ou plusieurs fichiers rpm

Dans ce cas, il suffit de passer le nom du fichier en argument à urpmi :

```
# urpmi mon_paquetage-version.i586.rpm
```

et le package sera mis à jour de la même manière que `rpm -U` l'aurait fait.

urpmi peut installer plusieurs paquetages d'un coup : par exemple, pour installer la dernière version de KDE que j'ai téléchargée, je me mets dans le répertoire où j'ai mis les rpm, et je tape :

```
# urpmi *.rpm
```

Pour installer (et non mettre à jour un package, par exemple pour ajouter un nouveau noyau en parallèle de l'ancien, ou bien une nouvelle librairie qui doit coexister avec l'ancienne), il faut utiliser l'option `-i` : `urpmi -i`

Désinstaller avec urpme

Pour désinstaller un paquetage, il faut utiliser **urpme** :

```
# urpme mplayer
```

désinstallera mplayer. S'il y a des paquetages qui en dépendent, urpme demandera s'il doit les désinstaller également. À utiliser avec précaution ;-))

Rechercher avec urpmf (et urpmq)

L'outil de choix pour faire de recherches dans les packages rpm et **urpmf**. Cela permet de rechercher non seulement dans les paquetages installés comme avec `rpm -q`, mais aussi des paquetages installables !

```
$ urpmf toto
```

renverra la liste de tous les paquetages qui contiennent le fichier toto.

urpmf dispose de nombreuses options pour faire des requêtes sur les différentes informations de la base de données des paquetages.

Par exemple :

Liste des paquetages du groupe "Games"

```
$ urpmf --group Games
```

Taille du paquetage "pingus"

```
$ urpmf --size pingus
```

Mettre à jour le système



```
pingus:size:1102629
```

Résumé du paquetage "pingus"

```
$ urpmpf --summary pingus
```

```
pingus:summary:Pingus - A free Lemmings clone
```

Voir man `urpmpf` pour de plus amples informations.

La commande **urpmq** vous permet d'autres types de requêtes :

Liste des paquetages dont le nom contient la chaîne "toto" :

```
$ urpmq toto
```

Sur quelle source se trouve le paquetage toto :

```
$ urpmq --archive toto
```

Quels paquetages dépendent du paquetage toto :

```
$ urpmq -d toto
```

Nom complet du fichier rpm du paquetage toto :

```
$ urpmq -r toto
```

Voir aussi man `urpmq`.

Mettre à jour le système

`urpm` vous permet de mettre à jour votre système avec les derniers correctifs de sécurité et les corrections de bugs que Mandrake publie régulièrement.

Pour ce faire, il faut avoir paramétré un miroir ftp de mises à jour de sécurité dans `urpm` (voir configuration), ou avec le Gestionnaire de Programmes (qui est bien pratique pour cette opération).

Si la source de mise à jour s'appelle `maj_secu`, il faut lancer la commandes suivante pour la mettre à jour :

```
# urpmi.update maj_secu
```

puis :

```
# urpmi --auto-select
```

`urpm` va alors lister les paquetages mis à jour et vous demander si vous souhaitez les installer (pour qu'il installe tout sans demander, pratique pour mettre la commande dans un cron, ajouter l'option `--auto`).

Si vous souhaitez que `urpm` ne mette pas à jour automatiquement certains paquetages, il suffit d'ajouter leur nom dans le fichier `/etc/urpmi/skip.list`.

Par exemple, pour empêcher `urpm` de mettre à jour automatiquement le kernel et la glibc, ajouter dans ce fichier :

```
kernel
glibc
```

Conclusion sur urpm

Les commandes `urpm` sont encore trop peu connues, et la maintenance de votre Mandrake sera bien plus aisée avec leur utilisation.

apt-get sur Redhat

Redhat dispose également d'outils pour faciliter la gestion des packages et la mise à jour du système. J'ai testé `rpm-get` que je n'ai pas trouvé satisfaisant. Par-contre le portage de `apt-get` sur redhat est véritablement une réussite. C'est donc cet utilitaire dont je parlerai concernant Redhat (pour l'article je travaille sur une Redhat 7.3)

Cette version de `apt-get` pour RPM, même si elle est critiquée parce qu'elle utilise `RPM` (contre `dpkg` pour Debian), est extrêmement efficace

Les principaux apports de `apt-get`:

- Prise en charge des dépendances pour l'installation et la désinstallation de packages
- Mise à jour complète de la distribution avec les dernières versions des packages existantes
- Comme pour `urpm`, installation des paquetages en donnant seulement un nom incomplet

Installation de apt-get

Les packages à installer :

Vous devez installer au moins le package `apt-get`. Il contient les binaires que nous allons utiliser par la suite. Pour ceux qui préféreraient utiliser une interface graphique, il en existe une, extrêmement conviviale, [synaptic](#).

La configuration :

Elle est très simple. Une fois les packages installés il vous suffit de taper `apt-get update`. Ceci va synchroniser votre base de données locale avec celle du serveur ftp. Cette commande sera à relancer régulièrement pour remettre à jour cette base de données (à insérer dans une crontab par exemple).

- `/etc/apt/sources.list` : contient la liste des dépôts pour la récupération des packages. Ces dépôts peuvent être des CD-ROM, un emplacement sur votre disque dur, des URL.
- `/etc/apt/apt.conf` : fichier de configuration de `apt-get`
- `/var/cache/apt/archives` : contient les packages qui ont été téléchargés pour être installés.

Commande supplémentaire :

La commande `apt-cdrom` vous permet de rajouter directement au fichier `sources.list` des dépôts sur CD-ROM. La syntaxe de la commande : `apt-cdrom --cdrom <point-de-montage> add`

Exemple : `root@pingu# apt-cdrom --cdrom /mnt/cdrom add`

Il vous reste alors à insérer successivement tous les CD en votre possession qui contiennent les packages Redhat.

Installer / Désinstaller / Mettre à jour des packages

Installer / Mettre à jour un ou plusieurs packages

- Pour installer un ou plusieurs packages, on utilisera la commande `apt-get` avec l'argument `install`.
Syntaxe : `apt-get install package1 [package2...]`
Exemple : `apt-get install mplayer` installera `mplayer` et les packages éventuellement nécessaires pour régler les dépendances.
- Vous pouvez également choisir d'installer des sources. Pour ce faire, il suffit d'utiliser l'argument `source`.
Exemple : `apt-get source galeon` vous permet de récupérer le RPM source de `galeon`.
- Pour mettre à jour un ou plusieurs packages, on utilisera la commande `apt-get` avec l'argument `upgrade`. Il est plus sûr au préalable d'exécuter la commande `apt-get update` pour mettre à jour la base de données locale et s'assurer que la mise à jour est réalisée avec la dernière version du dit package disponible.
Syntaxe : `apt-get upgrade package1 [package2...]`
Exemple : `apt-get upgrade mplayer` mettra à jour `mplayer` et les packages éventuellement nécessaires pour régler les dépendances.

Désinstaller avec `apt-get`

Désinstaller un ou plusieurs packages est aussi simple. Il suffit d'utiliser la commande `apt-get` suivie de l'argument `remove`.

Syntaxe : `apt-get remove package1 [package2...]`

Exemple : `apt-get remove mplayer` supprimera `mplayer` et les packages qui étaient dépendants de `mplayer` mais non utilisés par une autre application.

Mettre à jour le système

Enfin `apt-get` peut vous permettre également de mettre à jour la totalité de votre distribution. Au préalable, on exécutera là encore un `apt-get update` pou s'assurer de disposer des derniers packages. Puis il suffit de lancer la commande `apt-get dist-upgrade`. Et le tour est joué ;)

Vous disposez également de la commande `apt-get check`. Elle vous permet de vérifier que vous n'avez pas de dépendances non résolues sur votre système.

Rechercher des informations sur un package installé

Enfin la commande `apt-cache` vous permet d'obtenir un certain nombre d'informations sur les packages installés :

- Pour vérifier qu'un package est bien installé, il suffit d'utiliser l'argument `search`.
Syntaxe : `apt-cache search chaine_de_caractères` (la chaîne de caractères peut contenir des expressions régulières).
Exemple : vous voulez vérifier que `mplayer` est bien installé :

```
root@pingu# apt-cache search mplayer
mplayer - MPlayer, the Movie Player for Linux.
mplayer-skins - A collection of skins for MPlayer.
transcode - A Linux video stream processing utility.
```
- Pour vérifier les dépendances d'un package, on utilisera l'argument `depends`.
Syntaxe : `apt-cache depends chaine_de_caractères`
Exemple : vous voulez connaître les dépendances de `mplayer` :

```
root@pingu# apt-cache depends mplayer
mplayer
Depends: libdvdread
Depends: libdvdcss
libdvdcss2
Depends: gtk+
Depends: SDL
Depends: divx4linux
Depends: lame
Depends: libvorbis
Depends: lirc
Depends: libdv
Depends: aalib
Depends: arts
```