

# Ajouter la Capacité Multilib à Slackware pour l'Architecture x86\_64

transcription de la version française initiale écrite par <Patrick FONIO et Sébastien BALLET>

Cet article, qui est la traduction du travail réalisé par **Alien BOB**, contient les instructions nécessaires pour créer une vraie Slackware64 multilib.

Un système Linux multilib 64-bits peut faire tourner des applications 64-bits ainsi que celles encore fournies en 32-bits.

Le [Filesystem Hierarchy Standard](#) indiquant comment séparer clairement les programmes 64-bits et 32-bits sur un même système, avec la Slackware64 nous avons choisi d'adopter ce standard [x86\\_64](#). Elle est donc prête pour utiliser les bibliothèques 64-bits dans les répertoires /lib64 et /usr/lib64. C'est pourquoi je la nomme Slackware64 'multilib-ready' (même si les bibliothèques 32-bits se trouvent dans /lib et /usr/lib, Slackware64 ne contient aucun programme 32-bits).

Il y a toutefois une étape supplémentaire à franchir (par vous, l'utilisateur) pour que Slackware64 puisse être dénommée 'multilib-capable'.

On procède de la façon suivante .

1. Tout d'abord, nous avons besoin de :
  - \*glibc (c'est-à-dire : une glibc capable d'exécuter des binaires 32-bits et 64-bits)
  - \*gcc (c'est-à-dire : un gcc capable de générer des binaires 32-bits et 64-bits)
2. Ensuite, on récupère les bibliothèques de la Slackware 32-bits qu'on installe dans la Slackware64 à côté de leurs versions 64-bits, ce qui constitue la couche de compatibilité 32-bits.

Dès sa sortie, Slackware64 avait un avantage par rapport aux 'forks' 64-bits qui existaient par ailleurs. Ces 'forks' ajoutaient une couche de compatibilité 32-bits en recompilant bon nombre de paquets en binaires 32-bits. Slackware64 est une distribution constituée des versions 32-bits et 64-bits développées parallèlement. Ce qui signifie que vous n'avez pas à compiler des paquets 32-bits, vous avez juste à prendre les paquets 32-bits depuis l'arborescence Slackware-32. Voilà pourquoi nous n'avons pas ajouté un système multilib complet, les conditions sont en place mais demandent à l'utilisateur de faire le nécessaire s'il veut le multilib.

Dans un [prochain chapitre](#), j'expliquerai comment vous pouvez, à partir d'un paquet Slackware-32 (par exemple le paquet 'mesa') le réempaqueter en un paquet 'mesa-compat32' et l'installer tel quel sur la Slackware64.



Slackware pour les architectures x86\_64 (ou "Slackware64" en abrégé) est un système 64-bits « pur », mais qu'il est facile de faire évoluer en un système multilib. *Telle quelle, Slackware64 ne peut compiler et faire tourner que les binaires 64-bits.*

## Avantages d'un système multilib

Voici quelques exemples de programmes pour lesquels il faut un support multilib sur la Slackware64 car ils ne pourraient pas démarrer ni être compilés sans la couche de compatibilité 32-bits :

- [Wine](#)  
la plupart des programmes Windows sont en 32-bits, et pour les faire fonctionner sur Linux avec Wine, vous avez besoin de la version 32-bits de Wine.
- [VirtualBox](#)  
bien que ce soit (partiellement) de l'open source, il est nécessaire d'avoir les bibliothèques 32-bits sur Slack-64
- [Steam](#)  
la très populaire plate-forme de jeux doit encore disposer d'un [client 32-bits](#). La plupart des jeux sont disponibles en version 32-bits.
- [Skype](#), [Citrix client](#), ...  
ces programmes propriétaires sont fermés. Nous dépendons des développeurs pour que les versions 64-bits existent, ce qui n'est pas encore le cas à ce jour.

Heureusement, les développements 64-bits deviennent de plus en plus courants. Adobe a longtemps posé problème mais ils ont finalement livré une version 64-bits de leur plugin Flash. De même Sun (société maintenant intégrée au groupe Oracle) a publié une version 64-bits de leur plugin Java pour navigateur. Ces deux événements nous ont fortement incités à nous mettre au travail sur la Slackware64.

---

## Se procurer les paquets multilib

Vous pouvez télécharger un jeu complet de paquets et de scripts multilib depuis le site: <http://slackware.com/~alien/multilib/>

En plus de nombreux fichiers README (cet article de Wiki est fondamentalement une version augmentée de l'un de ces READMEs) vous trouverez un sous-répertoire pour chaque version de Slackware64 sous le chapitre « multilib ». Vous trouverez un autre répertoire nommé « source ». Celui-ci contient les sources (!!) et des scripts Slackbuilds.

Ce qui vous concerne vraiment -les paquets binaires- se trouve dans le répertoire « slackware\_n° de version » en dessous du répertoire principal. Chacun de ces répertoires contient aussi un sous-répertoire « slackware64-compat32 » où vous trouverez un ensemble de base de paquets Slackware 32-bits prêts à être installés sur votre Slackware64.

---

## Maintien de votre multilib à jour

Pour rester à jour, je vous conseille de surveiller le [ChangeLog \(flux RSS\)](#) que je maintiens pour mes paquets multilib. En principe je diffuserai les paquets *updated glibc and gcc* un jour ou deux après la mise à jour de gcc et glibc par Slackware.

Automatisation:

1. Consultez [compat32pkg](#) de Sébastien Ballet, qui automatise ces opérations, comme le fait slackpkg.
2. Si vous préférez slackpkg pour la gestion des paquets, alors cela vaut la peine de se pencher sur [slackpkg+](#), une extension à slackpkg qui gère les paquets que vous avez installés depuis des dépôts tiers - y compris multilib. Quand slackpkg+ est correctement configuré, la

maintenance de votre multilib se fait tout simplement ainsi:

```
# slackpkg update
# slackpkg upgrade multilib
# slackpkg install multilib
```

Cette dernière commande vous montrera si des paquets ont été ajoutés à la collection de ceux qui sont regroupés dans "compat32", comme llvm-compat32 et orc-compat32 récemment.

- Voici comment se présente une configuration typique - pour un ordinateur sur lequel tourne Slackware-current pointant vers le dépôt 'KDE testing' d'Alien BOB. L'option PKGS\_PRIORITY garantit que les paquets multilib de gcc et glibc ont la priorité sur les originaux Slackware. Le mot-clé "multilib" qui désigne le dépôt doit être le même mot que celui employé dans les commandes "slackpkg" ci-dessus. Le choix du mot "multilib" est arbitraire, il aurait tout aussi bien pu être "compat32", du moment que vous l'utilisez de façon cohérente.

Voici ce qu'il peut y avoir dans un fichier exemple de "/etc/slackpkg/slackpkgplus.conf":

```
SLACKPKGPLUS=on
VERBOSE=1
ALLOW32BIT=off
USEBL=1
WGETOPTS="--timeout=5 --tries=1"
GREYLIST=on
PKGS_PRIORITY=( multilib restricted alienbob ktown )
REPOPLUS=( slackpkgplus multilib restricted alienbob ktown )
MIRRORPLUS['multilib']=http://bear.alienbase.nl/mirrors/people/ali
en/multilib/current/
MIRRORPLUS['alienbob']=http://bear.alienbase.nl/mirrors/people/ali
en/sbrepos/current/x86_64/
MIRRORPLUS['restricted']=http://bear.alienbase.nl/mirrors/people/a
lien/restricted_sbrepos/current/x86_64/
MIRRORPLUS['ktown']=http://bear.alienbase.nl/mirrors/alien-kde/cur
rent/latest/x86_64/
MIRRORPLUS['slackpkgplus']=http://slakfinder.org/slackpkg+/
```

## Permettre le multilib sur Slackware64

### Vite fait, bien fait :

Ce chapitre contient les instructions essentielles pour ajouter la possibilité du multilib à votre Slackware-64. Si vous voulez comprendre le processus plus en profondeur, ou si vous avez besoin d'infos pour compiler des programmes 32-bits dans la Slackware-64, consultez les chapitres suivants. Notez que le "#" au début des commandes indique l'*invite de root*.

- Téléchargez les paquets depuis mon site (Je vous en ai indiqué l'URL [au paragraphe précédent](#), mais cet exemple utilise l'URL d'un miroir). Dans le cas où vous travaillez avec Slackware 14.2. faites:

```
# SLACKVER=14.2
# mkdir multilib
# cd multilib
# lftp -c "open http://bear.alienbase.nl/mirrors/people/alien/multilib/ ;
mirror -c -e ${SLACKVER}"
# cd ${SLACKVER}
```

- Mettez à jour vos paquets Slackware 64-bits de "gcc" et "glibc" vers mes versions multilib. Lancez la commande:

```
# upgradepkg --reinstall --install-new *.t?z
```

après vous être placé dans le répertoire où vous avez mis ces paquets.

Cette commande installera également un paquet complémentaire nommé "compat32-tools".

- Si vous avez aussi téléchargé le répertoire *slackware64-compat32* (ce qu'aura fait ma commande "lftp"), vous avez de la chance car j'ai déjà fait pour vous la conversion des paquets 32-bits! Il vous suffit de lancer:

```
# upgradepkg --install-new slackware64-compat32/*-compat32/*.t?z
```

qui va installer tous les paquets 32-bits Slackware (ou les mettre à jour si vous aviez déjà installé de plus anciens paquets multilib, par exemple lors du passage à une nouvelle Slackware). C'est tout!

- Si vous ne trouvez pas de sous-répertoire nommé *slackware64-compat32*, ou bien vous ne l'avez pas téléchargé ou bien le site miroir ne le fournissait pas. Dans ce cas, il vous faut convertir le paquet 32-bits vous-même. Ce n'est pas difficile du tout, ça ne prend que quelques minutes.
  - Le plus rapide, c'est d'avoir un répertoire local contenant les paquets 32-bits Slackware (c'est ce qu'on appelle un *miroir local*). Ceux qui ont acheté le DVD Slackware officiel peuvent simplement utiliser ce DVD: il est à double face et la Slackware 32-bits est sur une des faces. Pour la validité de cet exemple je considère que vous avez une arborescence locale Slackware en 32-bits disponible en `"/home/ftp/pub/slackware/slackware-14.2/slackware/"`. Il doit y avoir là, immédiatement accessibles dans ce répertoire, les sous-répertoires 'a', 'ap', 'd', 'l', 'n', 'x'. (Si vous avez monté un DVD Slackware, votre répertoire sera probablement `"/media/SlackDVD/slackware/"` mais je ne l'utiliserai pas dans les exemples de commandes ci-dessous).
  - Créez un nouveau répertoire vide (appelons le 'slackware64-compat32') et allons dedans, c'est à dire faisons en notre répertoire courant:

```
# mkdir slackware64-compat32 ; cd slackware64-compat32
```

- Lancez la commande suivante qui crée les paquets de compatibilité 32-bits, en prenant en entrée le répertoire officiel des paquets 32-bits Slackware:

```
# massconvert32.sh -i
/home/ftp/pub/slackware/slackware-14.2/slackware/
```

- L'étape précédente prend du temps. Quand c'est fini, installez les 90 Mo de paquets Slackware 32-bits fraîchement convertis qui viennent d'être créés dans les sous-

répertoires de votre *répertoire courant*:

```
# upgradepkg --install-new *-compat32/*.t?z
```

- C'est fait! Maintenant vous pouvez vous lancer dans le téléchargement, l'installation et l'utilisation de programmes 32-bits. Ce n'était pas si difficile, n'est-ce pas?

Si vous utilisez un gestionnaire de paquets comme *slackpkg* vous devrez ajouter tous les noms de paquets concernant *glibc* et *gcc* à sa liste noire de paquets. Si vous ne prenez pas cette précaution, vous risqueriez de voir votre gestionnaire de paquets remplacer par accident vos versions multilib par les versions d'origine Slackware en 'pur' 64-bits!

Si vous utilisez Slackware 13.37 ou plus récente, alors *slackpkg* interprète bien les expressions régulières dans le fichier de la liste noire. Dans ce cas, une seule ligne suffit dans `/etc/slackpkg/blacklist` pour faire exclure tous mes paquets par la liste noire (y compris les paquets multilib *gcc* et *glibc* et tous les paquets *compat32*:



```
[0-9]+alien  
[0-9]+compat32
```

Par contre, si vous employez l'extension de *slackpkg* nommée *slackpkg+* alors vous ne devez **surtout pas** mettre en liste noire ces paquets, parce que cela empêcherait *slackpkg+* de les gérer!

Si vous travaillez avec Slackware 13.1 ou plus récente, et si vous disposez du paquet *compat32-tools* pour cette édition, le script *massconvert32.sh* peut utiliser un serveur distant pour télécharger les paquets Slackware 32-bits, au lieu de nécessiter un serveur Slackware local ou un DVD. Employez le paramètre "-u" pour indiquer l'URL distante comme ceci:



```
# massconvert32.sh -u  
http://unserveur.org/chemin/vers/slackware-14.2/slackware
```

## Instructions détaillées

### Mise à niveau de *glibc* et *gcc*

Les paquets suivants remplacent (ne s'ajoutent pas aux...) les paquets standard de votre Slackware. Utilisez le programme 'upgradepkg' pour mettre à niveau vers la version multilib de *gcc* et *glibc* dont vous aurez besoin pour construire (*gcc*) et lancer (*glibc*), les programmes 32-bits utilisables sur votre ordinateur 64-bits avec Slackware64.

## Slackware64 13.0

- Le nécessaire pour le compilateur gcc:
  - gcc-4.3.3\_multilib-x86\_64-4alien.txz
  - gcc-g++-4.3.3\_multilib-x86\_64-4alien.txz
  - gcc-gfortran-4.3.3\_multilib-x86\_64-4alien.txz
  - gcc-gnat-4.3.3\_multilib-x86\_64-4alien.txz
  - gcc-java-4.3.3\_multilib-x86\_64-4alien.txz
  - gcc-objc-4.3.3\_multilib-x86\_64-4alien.txz
- Les bibliothèques GNU libc:
  - glibc-2.9\_multilib-x86\_64-3alien.txz
  - glibc-i18n-2.9\_multilib-x86\_64-3alien.txz
  - glibc-profile-2.9\_multilib-x86\_64-3alien.txz
  - glibc-solibs-2.9\_multilib-x86\_64-3alien.txz
  - glibc-zoneinfo-2.9\_multilib-noarch-3alien.txz

## Slackware64 13.1

- Le trousseau du compilateur gcc:
  - gcc-4.4.4\_multilib-x86\_64-1alien.txz
  - gcc-g++-4.4.4\_multilib-x86\_64-1alien.txz
  - gcc-gfortran-4.4.4\_multilib-x86\_64-1alien.txz
  - gcc-gnat-4.4.4\_multilib-x86\_64-1alien.txz
  - gcc-java-4.4.4\_multilib-x86\_64-1alien.txz
  - gcc-objc-4.4.4\_multilib-x86\_64-1alien.txz
- Les bibliothèques GNU libc:
  - glibc-2.11.1\_multilib-x86\_64-3alien.txz
  - glibc-i18n-2.11.1\_multilib-x86\_64-3alien.txz
  - glibc-profile-2.11.1\_multilib-x86\_64-3alien.txz
  - glibc-solibs-2.11.1\_multilib-x86\_64-3alien.txz
  - glibc-zoneinfo-2.11.1\_multilib-noarch-3alien.txz

## Slackware64 13.37

- Le trousseau du compilateur gcc:
  - gcc-4.5.2\_multilib-x86\_64-2alien.txz
  - gcc-g++-4.5.2\_multilib-x86\_64-2alien.txz
  - gcc-gfortran-4.5.2\_multilib-x86\_64-2alien.txz
  - gcc-gnat-4.5.2\_multilib-x86\_64-2alien.txz
  - gcc-java-4.5.2\_multilib-x86\_64-2alien.txz
  - gcc-objc-4.5.2\_multilib-x86\_64-2alien.txz
- Les bibliothèques GNU libc:
  - glibc-2.13\_multilib-x86\_64-7alien.txz
  - glibc-i18n-2.13\_multilib-x86\_64-7alien.txz
  - glibc-profile-2.13\_multilib-x86\_64-7alien.txz
  - glibc-solibs-2.13\_multilib-x86\_64-7alien.txz

## Slackware64 14.0

- Le trousseau du compilateur gcc:
  - gcc-g++-4.7.1\_multilib-x86\_64-1alien.txz
  - gcc-gfortran-4.7.1\_multilib-x86\_64-1alien.txz
  - gcc-gnat-4.7.1\_multilib-x86\_64-1alien.txz
  - gcc-go-4.7.1\_multilib-x86\_64-1alien.txz
  - gcc-java-4.7.1\_multilib-x86\_64-1alien.txz
  - gcc-objc-4.7.1\_multilib-x86\_64-1alien.txz
- Les bibliothèques GNU libc:
  - glibc-2.15\_multilib-x86\_64-9alien.txz
  - glibc-i18n-2.15\_multilib-x86\_64-9alien.txz
  - glibc-profile-2.15\_multilib-x86\_64-9alien.txz
  - glibc-solibs-2.15\_multilib-x86\_64-9alien.txz

## Slackware64 14.1

- Le trousseau du compilateur gcc:
  - gcc-4.8.2\_multilib-x86\_64-1alien.txz
  - gcc-g++-4.8.2\_multilib-x86\_64-1alien.txz
  - gcc-gfortran-4.8.2\_multilib-x86\_64-1alien.txz
  - gcc-gnat-4.8.2\_multilib-x86\_64-1alien.txz
  - gcc-go-4.8.2\_multilib-x86\_64-1alien.txz
  - gcc-java-4.8.2\_multilib-x86\_64-1alien.txz
  - gcc-objc-4.8.2\_multilib-x86\_64-1alien.txz
- Les bibliothèques GNU libc:
  - glibc-2.17\_multilib-x86\_64-10alien.txz
  - glibc-i18n-2.17\_multilib-x86\_64-10alien.txz
  - glibc-profile-2.17\_multilib-x86\_64-10alien.txz
  - glibc-solibs-2.17\_multilib-x86\_64-10alien.txz

## Slackware64 14.2

- Le trousseau du compilateur gcc:
  - gcc-5.3.0\_multilib-x86\_64-3alien.txz
  - gcc-g++-5.3.0\_multilib-x86\_64-3alien.txz
  - gcc-gfortran-5.3.0\_multilib-x86\_64-3alien.txz
  - gcc-gnat-5.3.0\_multilib-x86\_64-3alien.txz
  - gcc-go-5.3.0\_multilib-x86\_64-3alien.txz
  - gcc-java-5.3.0\_multilib-x86\_64-3alien.txz
  - gcc-objc-5.3.0\_multilib-x86\_64-3alien.txz
- Les bibliothèques GNU libc:
  - glibc-2.23\_multilib-x86\_64-2alien.txz
  - glibc-i18n-2.23\_multilib-x86\_64-2alien.txz
  - glibc-profile-2.23\_multilib-x86\_64-2alien.txz
  - glibc-solibs-2.23\_multilib-x86\_64-2alien.txz

## Slackware64 current

- Tant que vous ne voyez pas de répertoire distinct nommé “*current*”, vous pouvez utiliser les fichiers du répertoire de l'édition de la Slackware stable la plus récente.
- Le trousseau du compilateur gcc:
  - gcc-7.1.0\_multilib-x86\_64-2alien.txz
  - gcc-brig-7.1.0\_multilib-x86\_64-2alien.txz
  - gcc-g++-7.1.0\_multilib-x86\_64-2alien.txz
  - gcc-gfortran-7.1.0\_multilib-x86\_64-2alien.txz
  - gcc-gnat-7.1.0\_multilib-x86\_64-2alien.txz
  - gcc-go-7.1.0\_multilib-x86\_64-2alien.txz
  - gcc-objc-7.1.0\_multilib-x86\_64-2alien.txz
- Les bibliothèques GNU libc:
  - glibc-2.25\_multilib-x86\_64-3alien.txz
  - glibc-i18n-2.25\_multilib-x86\_64-3alien.txz
  - glibc-profile-2.25\_multilib-x86\_64-3alien.txz
  - glibc-solibs-2.25\_multilib-x86\_64-3alien.txz



Depuis le passage à gcc 7, il n'y a plus de paquet gcc - java parce qu'il n'est plus développé.



Le paquet glibc-zoneinfo n'est pas un élément de multilib, puisqu'il ne contient pas de code. Il vous faut installer le paquet glibc-zoneinfo de la Slackware de base.

## Pour toutes éditions de Slackware

Il y a un paquet supplémentaire que vous installerez en utilisant le programme “installpkg”. Le n° de version précis peut varier suivant l'édition de Slackware, mais on peut trouver le paquet dans le répertoire où se trouvent également les versions multilib de gcc et glibc:

- Le “32bit toolkit” (scripts qui facilitent la création de paquets 32-bits)
  - compat32-tools-3.5-noarch-4alien.tgz

## Ajout des bibliothèques Slackware 32-bits

La mise à niveau de glibc et gcc telle que décrite précédemment fait passer votre système de «*multilib-ready*» à «*multilib-enabled*». Tout ce que vous devez faire maintenant c'est installer les bibliothèques Slackware 32-bits afin que les programmes 32-bits que vous installerez ou compilerez trouvent les bibliothèques dont ils ont besoin. Mais il ne suffit pas de prendre les paquets 32-bits et les installer tels quels dans la Slackware64 pour au moins 2 raisons:

- tout d'abord, vous allez vous retrouver avec plusieurs paquets portant le même nom, source de confusion pour vous et aussi pour le gestionnaire de paquets 'slackpkg'.
- ensuite, si le paquet 32-bits que vous installez contient des binaires (tels /usr/bin/foo par

exemple) vous risquez d'écraser leur équivalent 64-bits, ce qui mettrait la pagaille dans votre système.

Il faudra encore veiller à ce que les fichiers inutiles/proscrits des paquets 32-bits que vous installez soient retirés. Ce qu'il vous faut, c'est un paquet 32-bits qui n'entre pas en conflit avec quoi que ce soit déjà présent dans votre Slackware 64-bits. D'où le nom "paquet de compatibilité 32-bits".

J'ai estimé que ce serait un gaspillage de bande passante si je créais moi-même les versions de compatibilité 32-bits des paquets Slackware. Après tout, vous avez probablement acheté le DVD Slackware 14.x et vous avez donc les deux versions de Slackware en 64-bits et 32-bits... ou bien

l'arborescence 32-bits de Slackware est disponible en téléchargement libre, bien sûr



Au lieu de celà, j'ai écrit quelques scripts (certaines parties de ces scripts sont l'œuvre de Fred Emmott du célèbre projet [Slamd64](#)) et je les ai regroupés en un paquet "*compat32-tools*". Ces scripts vous permettront d'utiliser le contenu de tout paquet 32-bits Slackware pour créer un nouveau paquet que vous pourrez installer en toute sécurité dans votre Slackware 64-bits.

Ce paquet "*compat32-tools*" a besoin de quelques explications.

Je vous invite à lire le '*README*' détaillé, du répertoire `/usr/doc/compat32-tools-*/`, il vous apportera une aide précieuse. Voici les trois scripts que le paquet installe:

- `/etc/profile.d/32dev.sh`

C'est le même script qui se trouve dans Slamd64. Il reconfigure l'environnement de votre shell pour qu'il soit plus simple pour vous de compiler des logiciels 32-bits (en préférant les compilateurs et bibliothèques 32-bits à leurs versions 64-bits).

- `convertpkg-compat32`

Ce script traite un paquet 32-bits Slackware et le convertit en un paquet '`-compat32`' que vous pouvez installer aisément (grâce à "`installpkg`") dans la Slackware64, à côté de la version 64-bits du même paquet logiciel. Par exemple: s'il vous faut des bibliothèques 32-bits disponibles dans le paquet mesa, prenez le paquet mesa dans Slackware 32-bits (`x/mesa-7.5-i486-1.txz`) et lancez

```
# convertpkg-compat32 -i /path/to/mesa-7.5-i486-1.txz
```

qui va créer un nouveau paquet nommé `mesa-compat32-7.5-x86_64-1compat32.txz`. Ce nouveau paquet (qui est créé dans votre répertoire `/tmp` sauf si vous indiquez une autre destination) est fondamentalement l'ancien paquet 32-bits, mais allégé de tout le superflu. Puisque le nom du paquet est changé (*mesa* devient *mesa-compat32*) il est maintenant possible d'installer ce nouveau paquet dans Slackware64 où il coexistera avec le paquet *mesa* 64-bits, sans écraser aucun fichier.

Le script laisse des fichiers temporaires dans le répertoire "`/tmp/package-<prgram>-compat32`" où vous pouvez les effacer sans risque.

- `massconvert32.sh`

Ce script contient une liste interne de ce que je considère comme le sous-ensemble essentiel des paquets Slackware 32-bits. Il emploie le script vu ci-dessus "`convertpkg-compat32`" pour traiter chaque paquet de cette liste interne, et les convertir en paquets '`-compat32`'.

Il vous suffit d'utiliser ce script une seule fois, par exemple comme ceci (dans l'exemple il est admis que votre DVD Slackware 32-bits est monté en `/mnt/dvd`):

```
# massconvert32.sh -i /mnt/dvd/slackware -d ~/compat32
```

Cela fait, vous obtenez environ 150 Mo de nouveaux paquets dans le répertoire récemment créé `~/compat32` (le nom du répertoire est arbitraire bien sûr, je l'ai choisi pour la clarté de cet exemple). Ces paquets comportent ce qui constitue la partie 32-bits de votre système Slackware multilib.

Vous les installerez au moyen de `"installpkg"`, et ils vous offriront une couche de compatibilité 32-bits vraiment complète pour la Slackware64:

```
# installpkg ~/compat32/*/*.t?z
```

Si vous effectuez une mise à jour depuis une version plus ancienne de ces paquets (par exemple à l'occasion de la mise à niveau de votre Slackware 64-bits vers une nouvelle édition) n'employez pas `"installpkg"` dans ce cas, bien sûr, mais plutôt `"upgradepkg --install-new"` :

```
# upgradepkg --install-new ~/compat32/*/*.t?z
```

Le paramètre `"--install-new"` est requis pour installer les paquets `compat32` qui ont été ajoutés au fil des éditions successives.

En installant les paquets `compat32` vous remarquerez que quelques uns vont indiquer des erreurs dues à certains fichiers manquants dans `/etc`. C'est "pensé à la conception", et on peut ignorer ces erreurs. Ces messages apparaissent parce que dans `/etc` des fichiers sont retirés d'un paquet `"-compat32"` pendant la conversion (sauf pour `pango` et `gtk+2`). Je considère que les fichiers dans `/etc` auront déjà été installés par les paquets 64-bits d'origine.

Un exemple de ces "erreurs" pour le paquet `cups-compat32`:



```
Executing install script for cups-compat32-1.3.11-x86_64-1.txz.  
install/doinst.sh: line 5: [: too many arguments  
cat: etc/cups/interfaces: Is a directory  
cat: etc/cups/ppd: Is a directory  
cat: etc/cups/ssl: Is a directory  
cat: etc/cups/*.new: No such file or directory  
cat: etc/dbus-1/system.d/cups.conf.new: No such file or  
directory  
chmod: cannot access `etc/rc.d/rc.cups.new': No such file or  
directory  
cat: etc/rc.d/rc.cups.new: No such file or directory  
Package cups-compat32-1.3.11-x86_64-1.txz installed.
```



Si vous envisagiez d'utiliser le script `convertpkg-compat32` pour convertir un paquet **non-Slackware** en un paquet `-compat32`, je dois fermement vous mettre en garde contre cette idée. Le script est écrit dans le seul but d'obtenir les versions 32-bits des binaires/bibliothèques Slackware64 disponibles dans une configuration multilib. De ce fait, le script va enlever plusieurs éléments qui font partie du paquet



32-bits original - éléments que l'on considère présents car installés par la version 64-bits du paquet.

Dans presque tous les cas où, après téléchargement d'un paquet non-Slackware 32-bits dans la Slackware64, le mieux est d'en trouver les sources pour construire une version 64-bits de ce paquet. Une autre solution est d'*installer le paquet original* 32-bits, au lieu d'essayer de "le convertir", puis de l'activer à partir de la ligne de commande pour trouver les bibliothèques manquantes 32-bits qu'il vous faut encore extraire d'un paquet Slackware officiel.

## Utiliser les programmes 32-bits

Une fois que votre système a été préparé comme indiqué ci-dessus, il est très facile de faire tourner un logiciel compilé pour architecture 32-bits. Il suffit de le télécharger, l'installer et le lancer!

Parfois, vous rencontrerez peut-être un programme nécessitant une bibliothèque Slackware 32-bits particulière de laquelle vous ne disposez pas à ce moment. Dans ce cas, trouvez dans quel paquet Slackware 32-bits il y a cette bibliothèque manquante. Utilisez le script "*convertpkg-compat32*" pour convertir ce paquet Slackware pour architecture 32-bits et installez le paquet résultant 32-bits doté de la "*compatibilité*" dans Slackware64.

## Compiler des programmes 32-bits

Si vous devez compiler un programme 32-bits (wine et grub sont deux exemples de programmes open source livrés uniquement en version 32-bits) configurez d'abord l'environnement de votre shell pour root en lançant la commande:

```
# . /etc/profile.d/32dev.sh
```

Remarquez le point en début de ligne - il fait vraiment partie de la commande! L'emploi du point équivaut à la commande 'source'.

L'action de cette commande est de changer ou créer plusieurs variables d'environnement. Ce qui a pour effet de donner la priorité aux versions binaires 32-bits sur les binaires 64-bits quand vous compilez du code source - vous effectuerez ainsi une compilation 32-bits. L'effet de cette commande cessera au moment de votre déconnexion du shell 'root' -du super utilisateur-. Dans cet environnement modifié, vous pouvez employer les SlackBuilds standard pour construire des paquets 32-bits pour Slackware64. Il y a deux choses à ne pas perdre de vue:

1. Vous devez renseigner la variable ARCH avec 'i486' parce que même sur votre ordinateur 'x86\_64' vous êtes en train de compiler un programme pour architecture 32-bits! C'est à cause du *triplet* "\$ARCH-slackware-linux" qui normalement est employé dans la commande "configure".
  1. Il y a une exception pour la compilation du paquet "wine" pour lequel il faut 'ARCH=x86\_64' parce que vous installerez ce paquet directement dans votre ordinateur multilib sans le convertir en un paquet 'compat32'.
2. L'installation de paquets 32-bits que vous avez compilés comme vu au 1., sera possible sur votre Slackware64-multilib à condition de les convertir en paquets 'compat32':

```
# convertpkg -compat32 -i /path/to/your/fresh/foo-VERSION-i486-BUILD.tgz
# upgradepkg --install-new /tmp/foo-compat32-VERSION-x86_64-
BUILDcompat32.txz
```

## Mises en garde

- Quand vous aurez installé les paquets “-compat32”, vous devrez peut-être réinstaller les pilotes binaires *Nvidia* ou *Ati* pour gérer votre carte vidéo dans X.Org. Ces paquets de pilotes du matériel contiennent à la fois les bibliothèques 64-bits et 32-bits pour être efficaces au maximum sur un OS 64-bits multilib. Si vous avez installé les fichiers des pilotes pour les deux architectures, le paquet “*mesa-compat32*” écrasera certains fichiers de la bibliothèque 32-bits.

Par contre, si au départ vous aviez installé *seulement* les bibliothèques du pilote 64-bits pour votre carte Nvidia/Ati, il est recommandé après l'installation des paquets *multilib*, de réinstaller le paquet du driver binaire. Cette fois, choisissez d'installer aussi les fichiers du pilote 32-bits.

Les applications graphiques 32-bits que vous allez faire tourner dans votre installation multilib auront besoin de ces bibliothèques pour pilotes 32-bits. Il y aura sûrement des plantages si vous n'installez pas les bons fichiers.

- Si vous souhaitez compiler votre noyau 64-bits vous-même, assurez vous de choisir en option de compilation l'aptitude à l'émulation 32-bits pour lui, sinon multilib échouerait mystérieusement. Il vous faudra cet élément de configuration du noyau:

**CONFIG\_IA32\_EMULATION**

## Paquets convertis par massconvert32.sh

Voici la liste des paquets convertis en leur version “-compat32” par le script `massconvert32.sh`. Notez que plusieurs de ces paquets n'appartiennent pas à Slackware 13.0 ou 13.1, ils ont été ajoutés dans une version ultérieure de Slackware ce qui fait qu'ils provoquent un message d'erreur “\* **FAIL: package 'package\_name' was not found!**” quand vous lancez le script dans une version plus ancienne. La situation complémentaire se manifeste également - certains paquets ont été *retirés* des versions plus récentes de Slackware, ce qui déclenche le message “\* **FAIL: package 'package\_name' was not found!**”. Ne vous inquiétez pas pour ça.

```
# Les ensembles A/ :
```

```
aaa_elflibs
attr
bzip2
cups
cxxlibs
dbus
e2fsprogs
eudev
libgudev
openssl-solibs
```

```
udev
util-linux
xz

# Les ensembles AP/ :

cups
cups-filters
flac
mariadb
mpg123
mysql
sqlite

# Les ensembles D/ :

libtool
llvm
opencl-headers

# Les ensembles L/ :

SDL2
alsa-lib
alsa-oss
alsa-plugins
atk
audiofile
cairo
dbus-glib
elfutils
esound
expat
ffmpeg
fftw
freetype
fribidi
gamin
gc
gdk-pixbuf2
giflib
glib2
gmp
gnome-keyring
gtk+2
gst-plugins-base
gst-plugins-base0
gst-plugins-good
gst-plugins-good0
gst-plugins-libav
gstreamer
```

gstreamer0  
hal  
harfbuzz  
icu4c  
jasper  
json-c  
**lame**  
lcms  
lcms2  
libaio  
libart\_lgpl  
libasyncns  
libclc  
libedit  
libelf  
libexif  
libffi  
libglade  
libgphoto2  
libidn  
libieee1284  
libjpeg  
libjpeg-turbo  
libmng  
libmpc  
libnl3  
libnotify  
libogg  
libpcap  
libpng  
libsamplerate  
libsndfile  
libtasn1  
libtermcap  
libtiff  
libunistring  
libusb  
libvorbis  
libxml2  
libxslt  
lzo  
ncurses  
ocl-icd  
openjpeg  
orc  
pango  
popt  
pulseaudio  
python-six  
qt  
readline

```
sbcl
SDL
seamonkey-solibs
speexdsp
startup-notification
svgalib
v4l-utils
zlib

# Les ensembles N/ :

curl
cyrus-sasl
gnutls
libgcrypt
libgpg-error
libtirpc
nettle
openldap-client
openssl
p11-kit
samba

# Les ensembles X/ :

fontconfig
freeglut
glew
glu
libFS
libICE
libSM
libX11
libXScrnSaver
libXTrap
libXau
libXaw
libXcomposite
libXcursor
libXdamage
libXdmcp
libXevie
libXext
libXfixes
libXfont
libXfont2
libXfontcache
libXft
libXi
libXinerama
libXmu
```

```
libXp
libXpm
libXprintUtil
libXrandr
libXrender
libXres
libXt
libXtst
libXv
libXvMC
libXxf86dga
libXxf86misc
libXxf86vm
libdmx
libdrm
libepoxy
libfontenc
libinput
libfontenc
libpciaccess
libva
libva-intel-driver
libvdpau
libxcb
libxshmfence
mesa
pixmap
vulkan-sdk
xcb-util
```

```
# Les ensembles XAP/ :
```

```
sane
```

## Miroirs de téléchargement du système multilib

Vous pouvez télécharger les paquets multilib depuis (au moins) ces adresses:

- <http://slackware.com/~alien/multilib/>
- <http://bear.alienbase.nl/mirrors/people/alien/multilib/>
- <http://slackware.org.uk/people/alien/multilib/>
- <http://alien.slackbook.org/slackware/multilib/>
- <http://slackbuilds.org/mirror/alien/multilib/>

## Outils compatibles venant d'autres contributeurs

- Sébastien Ballet a écrit un outil nommé *compat32pkg*. Sur [son site](#) il y a *compat32pkg* disponible au téléchargement ainsi que la documentation complète indiquant comment l'utiliser

avec la Slackware64.

En voici une citation:

*“Compat32pkg est un outil automatisé qui offre tout ce qu'il faut pour gérer (convertir, installer, mettre à jour, supprimer) la structure 32-bits du multilib d'AlienBob pour slackware-64, et tous les paquets 32-bits qui proviennent de Slackware-32 que les utilisateurs peuvent être amenés à faire tourner dans un environnement 64-bits, comme firefox, seamonkey, jre,...”*

- Il y a aussi [slackpkg+](#), écrit par Matteo Rossini (surnommé zerouno) avec des contributions de Sébastien Ballet (parmi d'autres). C'est un greffon au logiciel [slackpkg](#) qui donne la possibilité d'installer des paquets venant de dépôts externes (de tierce origine) Slackware non officiels. Il est bien conçu et documenté pour ajouter à votre Slackware 64-bits l'aptitude au multilib et la maintenir à jour.

## Traductions

- Bruno Russo a traduit cet article en portugais (Brésil):  
[http://www.brunorusso.eti.br/slackware/doku.php?id=multilib\\_para\\_o\\_slackware\\_x86\\_64](http://www.brunorusso.eti.br/slackware/doku.php?id=multilib_para_o_slackware_x86_64)
- Mehdi Esmaeelpour a traduit cet article en persan:  
<http://www.slack-world.com/index.php/articles/43-general-system/85-multilib-slackware64>
- Patrick FONIO et Sébastien BALLETT ont traduit cet article en français:  
<http://wiki.slackware-fr.org/avance:articles:slackware64-multilib>

## Remerciements

- Je dois grandement remercier Fred Emmott, le créateur de Slamd64, le premier système 64-bits non officiel dérivé de Slackware. Bien que Slackware64 ne soit pas basée sur le travail de Fred, j'ai cependant appris beaucoup concernant la constitution et l'agencement de la partie 32-bits d'un système Linux multilib grâce à ce qu'il a écrit dans Slamd64.  
Notez que Slamd64 employait des paquets multilib de gcc/glibc séparés pour 64-bits et 32-bits. Toutefois, je crois qu'il est plus clair de conserver l'unité de ces paquets multilib qui sont essentiels. J'ai suivi le concept déjà adopté pour le paquet *binutils* de Slackware64, dont les aptitudes multilib en 64-bits et en 32-bits sont associées en un seul paquet.
- Cross Linux From Scratch.  
Le Wiki CLFS (<http://trac.cross-lfs.org/wiki/read#ReadtheCrossLinuxFromScratchBookOnline>) est 'à lire absolument' si vous voulez comprendre comment adapter Linux à une nouvelle architecture. J'en ai retenu plusieurs idées, concepts et correctifs quand j'ai construit Slackware64 à partir de zéro, puis encore quand j'ai créé mes paquets multilib gcc/glibc 'ex nihilo' (depuis table rase), mon README au sujet de ce multilib-from-scratch se trouve dans le répertoire `./source`.

Régalez vous!

Eric

## Sources

- L'article original, écrit par Eric Hameleers, se trouve à <http://alien.slackbook.org/dokuwiki/doku.php?id=slackware:multilib>
- Traduit de l'anglais par — *Cedric M. 2015/09/08 18:07*
  - \* *Contributions de — P-M Averseng en 2016*

From:

<https://docs.slackware.com/> - **SlackDocs**

Permanent link:

<https://docs.slackware.com/fr:slackware:multilib>

Last update: **2017/09/22 07:15 (UTC)**

