Utilitaires Bacula

It comes in the night and sucks the essence from your computers.

Kern Sibbald

5 mai 2010
Ce manuel couvre la version 5.0.2 (28 April 2010) de Bacula.

Table des matières

Chapitre 1

Utilitaires de Volumes

Ce document décrit les programmes utilitaires écrits pour assister les utilisateurs et les développeurs de Bacula dans la gestion des volumes externes à Bacula.

1.1 Spécification du fichier de configuration

A partir de la version 1.27, chacun des programmes suivants nécessite d'avoir un fichier de configuration valide du Storage deamon. (en fait, seules les définitions de ressources **Device** sont utilisées par ces programmes). Ceci permet aux programmes de trouver les paramètres de configuration de votre périphérique d'archivage (en général un lecteur de bandes). Par défaut, ils lisent **bacula-sd.conf** dans le répertoire courant, mais vous pouvez spécifier un fichier de configuration différent avec l'option **-c**.

1.2 Spécification d'un nom de périphérique pour une bande

Chacun de ses programmes nécessites un **device-name** où le Volume peut être trouvé. Dans le cas des bandes, il s'agit du nom du périphérique comme **/dev/nst0** ou **/dev/rmt/0ubn** suivant le système d'exploitation. Pour que le programme fonctionne, il doit trouver un nom identique dans la ressource Device du fichier de configuration. Voir plus bas pour comment préciser les noms de Volume.

Si Bacula est en cours de fonctionnement et que vous souhaitez utiliser un de des programmes, vous devez soit arrêter le Storage daemon ou bien **démonter** (**unmount**) chaque lecteur de bandes que vous voulez utiliser (sinon il sera "**busy**", car en cours d'utilisation par Bacula).

1.3 Spécification d'un nom de périphérique pour un fichier

Si vous essayez de lire ou écrire un fichier d'archive au lieu d'une bande, le **device-name** sera le chemin complet d'accès au fichier d'archive (nom du fichier compris). Le nom du fichier est utilisé pour générer le nom du Volume, et le chemin résultant devra correspondre au contenu du fichier de configuration. En résumé, le chemin est équivalent au nom du périphérique de sauvegarde, et le nom du fichier au nom de volume.

1.4 Spécification des Volumes

En général, vous devez spécifier le nom du Volume pour chacun des programmes ci-dessous (à l'exception de **btape**). La meilleure méthode est spécifier un fichier de **bootstrap** en ligne de commande avec l'option **-b**. Vous allez ensuite devoir spécifier le ou les noms de Volume dans le fichier de boostrap. Supposez par exemple

que vous souhaitiez lire les bandes **tape1** et **tape2** : il faut tout d'abord créer un fichier de **bootstrap** appelé **list.bsr** qui contient :

```
Volume=test1|test2
```

où chaque nom de Volume est séparé par une barre verticale. Tapez ensuite la commande suivante :

```
./bls -b list.bsr /dev/nst0
```

Dans le cas où les Volumes Bacula sont des fichiers, vous pouvez juste concaténer les noms de volumes ainsi :

```
./bls /tmp/test1\|test2
```

L'antislash (\) est nécessaire ici comme caractère d'échappement pour le shell, afin de pouvoir utiliser la barre verticale (|).

Pour terminer, s'il vous semble qu'il est compliqué de spécifier les noms de Volumes avec un fichier de bootstrap, il est possible d'utiliser l'option **-V** (pour tous les programmes sauf **bcopy** pour spécifier un ou plusieurs noms de Volume, séparés par une barre verticale (|), comme par exemple :

```
./bls -V Vol001 /dev/nst0
```

Vous pouvez également utiliser un astérisque (*) pour indiquer que le programme doit accepter n'importe quel volume, comme par exemple :

```
./bls -V* /dev/nst0
```

1.5 bls

bls est utilisé pour faire un affichage de type listing comme **ls** d'une bande ou d'un fichier **Bacula**. Il est utilisé ainsi :

```
Usage: bls [options] <device-name>
       -b <file>       specify a bootstrap file
       -c <file>       specify a config file
       -d <level>      specify debug level
       -e <file>       exclude list
       -i <file>       include list
       -j              list jobs
       -k              list blocks
    (no j or k option) list saved files
       -L              dump label
       -p              proceed inspite of errors
       -v              be verbose
       -V              specify Volume names (separated by |)
       -?              print this message
```

Pour afficher le contenu d'une bande, il peut être utilisé ainsi :

```
./bls -V Volume-name /dev/nst0
```

Ou pour afficher le contenu d'un fichier :

```
./bls /tmp/Volume-name
or
./bls -V Volume-name /tmp
```

Attention, dans le cas d'un fichier, le nom du Volume devient le nom du fichier : dans l'exemple ci-dessus, vous devez remplacer **Volume-name** par votre nom du Volume (fichier).

Si vous ne précisez aucune option, **bls** affichera l'équivalent d'un **ls -l** pour chaque fichier de la bande. En utilisant les options indiquées plus haut, il est possible de n'afficher que les enregistrements de Job, les blocs de la bande, etc. Par exemple :

```
./bls /tmp/File002
bls: butil.c:148 Using device: /tmp
drwxrwxr-x   3 k  k  4096 02-10-19 21:08  /home/kern/bacula/k/src/dird/
drwxrwxr-x   2 k  k  4096 02-10-10 18:59  /home/kern/bacula/k/src/dird/CVS/
-rw-rw-r--   1 k  k    54 02-07-06 18:02  /home/kern/bacula/k/src/dird/CVS/Root
-rw-rw-r--   1 k  k    16 02-07-06 18:02  /home/kern/bacula/k/src/dird/CVS/Repository
-rw-rw-r--   1 k  k  1783 02-10-10 18:59  /home/kern/bacula/k/src/dird/CVS/Entries
-rw-rw-r--   1 k  k 97506 02-10-18 21:07  /home/kern/bacula/k/src/dird/Makefile
-rw-r--r--   1 k  k  3513 02-10-18 21:02  /home/kern/bacula/k/src/dird/Makefile.in
-rw-rw-r--   1 k  k  4669 02-07-06 18:02  /home/kern/bacula/k/src/dird/README-config
-rw-r--r--   1 k  k  4391 02-09-14 16:51  /home/kern/bacula/k/src/dird/authenticate.c
-rw-r--r--   1 k  k  3609 02-07-07 16:41  /home/kern/bacula/k/src/dird/autoprune.c
-rw-rw-r--   1 k  k  4418 02-10-18 21:03  /home/kern/bacula/k/src/dird/bacula-dir.conf
...
-rw-rw-r--   1 k  k    83 02-08-31 19:19  /home/kern/bacula/k/src/dird/.cvsignore
bls: Got EOF on device /tmp
84 files found.
```

1.5.1 Affichage des jobs

Si vous affichez le contenu d'un volume pour trouver quels Jobs restaurer, l'option **-j** devrait vous fournir tout ce que vous voulez tant que vous n'avez pas plusieurs clients. Par exemple :

```
./bls -j -V Test1 -c stored.conf DDS-4
bls: butil.c:258 Using device: "DDS-4" for reading.
11-Jul 11:54 bls: Ready to read from volume "Test1" on device "DDS-4" (/dev/nst0).
Volume Record: File:blk=0:1 SessId=4 SessTime=1121074625 JobId=0 DataLen=165
Begin Job Session Record: File:blk=0:2 SessId=4 SessTime=1121074625 JobId=1 Level=F Type=B
Begin Job Session Record: File:blk=0:3 SessId=5 SessTime=1121074625 JobId=5 Level=F Type=B
Begin Job Session Record: File:blk=0:6 SessId=3 SessTime=1121074625 JobId=2 Level=F Type=B
Begin Job Session Record: File:blk=0:13 SessId=2 SessTime=1121074625 JobId=4 Level=F Type=B
End Job Session Record: File:blk=0:99 SessId=3 SessTime=1121074625 JobId=2 Level=F Type=B
   Files=168 Bytes=1,732,978 Errors=0 Status=T
End Job Session Record: File:blk=0:101 SessId=2 SessTime=1121074625 JobId=4 Level=F Type=B
   Files=168 Bytes=1,732,978 Errors=0 Status=T
End Job Session Record: File:blk=0:108 SessId=5 SessTime=1121074625 JobId=5 Level=F Type=B
   Files=168 Bytes=1,732,978 Errors=0 Status=T
End Job Session Record: File:blk=0:109 SessId=4 SessTime=1121074625 JobId=1 Level=F Type=B
   Files=168 Bytes=1,732,978 Errors=0 Status=T
11-Jul 11:54 bls: End of Volume at file 1 on device "DDS-4" (/dev/nst0), Volume "Test1"
11-Jul 11:54 bls: End of all volumes.
```

montre une sauvegarde complète suivie de deux incrémentales. (? ? ? ? FIXME ? ? ? ? ?)

L'ajout de l'option **-v** affichera quasiment toutes les informations disponibles pour chaque enregistrement :

1.5.2 Affichage des blocs

En temps normal, sauf dans le but de débugguer (? ? ? FIXME ? ? ? déverminer), vous n'aurez pas besoin d'afficher les blocs (unité de stockage élémentaire des données de Bacula sur le Volume). Vous pouvez toujours le faire ainsi :

```
./bls -k /tmp/File002
bls: butil.c:148 Using device: /tmp
Block: 1 size=64512
```

```
Block: 2 size=64512
...
Block: 65 size=64512
Block: 66 size=19195
bls: Got EOF on device /tmp
End of File on device
```

En ajoutant l'option **-v**, vous obtiendrez encore plus d'informations, ce qui pourrait être utile pour savoir quelles sessions ont été écrites sur ce Volume.

```
./bls -k -v /tmp/File002
Volume Label:
Id                : Bacula 0.9 mortal
VerNo             : 10
VolName           : File002
PrevVolName       :
VolFile           : 0
LabelType         : VOL_LABEL
LabelSize         : 147
PoolName          : Default
MediaType         : File
PoolType          : Backup
HostName          :
Date label written: 2002-10-19 at 21:16
Block: 1 blen=64512 First rec FI=VOL_LABEL SessId=1 SessTim=1035062102 Strm=0 rlen=147
Block: 2 blen=64512 First rec FI=6 SessId=1 SessTim=1035062102 Strm=DATA rlen=4087
Block: 3 blen=64512 First rec FI=12 SessId=1 SessTim=1035062102 Strm=DATA rlen=5902
Block: 4 blen=64512 First rec FI=19 SessId=1 SessTim=1035062102 Strm=DATA rlen=28382
...
Block: 65 blen=64512 First rec FI=83 SessId=1 SessTim=1035062102 Strm=DATA rlen=1873
Block: 66 blen=19195 First rec FI=83 SessId=1 SessTim=1035062102 Strm=DATA rlen=2973
bls: Got EOF on device /tmp
End of File on device
```

Armé du SessionId et du SessionTime, vous pouvez extraire à peu près n'importe quoi du Volume.

Si vous souhaitez en savoir encore plus, ajoutez un second **-v** à la ligne de commande pour avoir un dump de chaque enregistrement de chaque bloc.

```
./bls -k -v -v /tmp/File002
bls: block.c:79 Dump block  80f8ad0: size=64512 BlkNum=1
               Hdrcksum=b1bdfd6d cksum=b1bdfd6d
bls: block.c:92    Rec: VId=1 VT=1035062102 FI=VOL_LABEL Strm=0 len=147 p=80f8b40
bls: block.c:92    Rec: VId=1 VT=1035062102 FI=SOS_LABEL Strm=-7 len=122 p=80f8be7
bls: block.c:92    Rec: VId=1 VT=1035062102 FI=1 Strm=UATTR len=86 p=80f8c75
bls: block.c:92    Rec: VId=1 VT=1035062102 FI=2 Strm=UATTR len=90 p=80f8cdf
bls: block.c:92    Rec: VId=1 VT=1035062102 FI=3 Strm=UATTR len=92 p=80f8d4d
bls: block.c:92    Rec: VId=1 VT=1035062102 FI=3 Strm=DATA len=54 p=80f8dbd
bls: block.c:92    Rec: VId=1 VT=1035062102 FI=3 Strm=MD5 len=16 p=80f8e07
bls: block.c:92    Rec: VId=1 VT=1035062102 FI=4 Strm=UATTR len=98 p=80f8e2b
bls: block.c:92    Rec: VId=1 VT=1035062102 FI=4 Strm=DATA len=16 p=80f8ea1
bls: block.c:92    Rec: VId=1 VT=1035062102 FI=4 Strm=MD5 len=16 p=80f8ec5
bls: block.c:92    Rec: VId=1 VT=1035062102 FI=5 Strm=UATTR len=96 p=80f8ee9
bls: block.c:92    Rec: VId=1 VT=1035062102 FI=5 Strm=DATA len=1783 p=80f8f5d
bls: block.c:92    Rec: VId=1 VT=1035062102 FI=5 Strm=MD5 len=16 p=80f9668
bls: block.c:92    Rec: VId=1 VT=1035062102 FI=6 Strm=UATTR len=95 p=80f968c
bls: block.c:92    Rec: VId=1 VT=1035062102 FI=6 Strm=DATA len=32768 p=80f96ff
bls: block.c:92    Rec: VId=1 VT=1035062102 FI=6 Strm=DATA len=32768 p=8101713
bls: block.c:79 Dump block  80f8ad0: size=64512 BlkNum=2
               Hdrcksum=9acc1e7f cksum=9acc1e7f
bls: block.c:92    Rec: VId=1 VT=1035062102 FI=6 Strm=contDATA len=4087 p=80f8b40
bls: block.c:92    Rec: VId=1 VT=1035062102 FI=6 Strm=DATA len=31970 p=80f9b4b
bls: block.c:92    Rec: VId=1 VT=1035062102 FI=6 Strm=MD5 len=16 p=8101841
...
```

1.6 bextract

Si vous devez utiliser **bextract**, vous avez sans doute fait quelque chose de travers. Par exemple, Si vous essayez de restaurer un fichier mais que vous rencontrez des problèmes, voyez la section Restoring When Things Go Wrong du chapitre Restauration de ce manuel.

Une restauration se fait normalement en exécutant un Job **Restore** depuis la **Console**. Toutefois, **bextract** est utile pour extraire un seul fichier (ou une liste de fichiers) d'une bande ou d'un fichier Bacula. En fait, **bextract** peut être un outil très utile pour restaurer des fichiers sur système vierge en supposant qu'il démarre, que vous disposes d'un binaire statique de **bextract** et d'un fichier de **bootstrap** approprié.

Il faut garder à l'esprit les limitations actuelles de bextract :

1. Il ne peut pas restaurer les listes de contrôle d'accès (fr :LCA, en :ACL) qui ont été sauvegardées en plus des données.
2. Il ne peut pas restaurer les fichiers cryptés.
3. La longueur de la ligne de commande est assez limitée, ce qui signifie que vous ne pourrez pas saisir un grand nombre de Volumes. Si vous avez besoin d'indiquer plus de Volumes que ce que la ligne de commande supporte, utilisez un fichier de bootstrap (voir plus bas)

bextract est invoqué de la façon suivante :

```
Usage: bextract [-d debug_level] <device-name> <directory-to-store-files>
       -b <file>       specify a bootstrap file
       -dnn            set debug level to nn
       -e <file>       exclude list
       -i <file>       include list
       -p              proceed inspite of I/O errors
       -V              specify Volume names (separated by |)
       -?              print this message
```

où **device-name** est l'Archive Device (nom brut ou nom de fichier complet) du périphérique à lire, et **directory-to-store-files** est un chemin qui sera préfixé à tous les fichiers restaurés.

NOTE : sur les machines Windows, si vous indiquez un préfixe comme d :/tmp, tout fichier qui aurait du être restauré dans **c :/My Documents** sera restauré dans **d :/tmp/My Documents**. En clair, le nom du disque initial est supprimé lors de la restauration. Si aucun préfixe n'est précisé, le fichier est restauré sur son disque d'origine.

1.6.1 Extraire avec des listes Include ou Exclude

En utilisant l'option **-e**, vous pouvez indiquer un fichier contenant une liste de fichiers à exclure. Des caractères joker peuvent être utilisés dans la liste d'exclusion. Cette option est normalement utilisée en conjonction avec l'option **-i** (voir plus bas). Les deux options **-e** et **-i** peuvent être utilisées en même temps avec l'option **-b**. Les filtres de bootstrap sont appliqués en premiers, suivis de la liste Include, puis de la liste Exclude.

De même, et c'est sans doute plus important, en utilisant l'option **-i**, vous pouvez indiquer une fichier contenant une liste de fichiers et de répertoires (un fichier/répertoire par ligne) à inclure dans la restauration. La liste doit contenir les noms de fichiers complets (avec le chemin). Si vous indiquez seulement un nom de répertoire, tous les fichiers et les sous-répertoires de ce chemin seront restaurés. Si vous spécifiez une ligne ne contenant qu'un nom de fichier seul (ie. **my-file.txt**), il ne sera sans doute pas extrait car vous n'avez pas précisé le chemin complet.

Si par exemple le fichier **include-list** contient :

```
/home/kern/bacula
/usr/local/bin
```

Alors la commande :

```
./bextract -i include-list -V Volume /dev/nst0 /tmp
```

va restaurer de l'archive Bacula **/dev/nst0** tous les fichiers et répertoires sauvegardés de **/home/kern/bacula** et **/usr/local/bin**. Les fichiers seront restaurés avec leur chemin d'accès complet, mais sous le répertoire **/tmp** (i.e. /tmp/home/kern/bacula/... and /tmp/usr/local/bin/...).

1.6.2 Extraire avec un fichier Bootstrap

L'option **-b** est utilisée pour spécifier un fichier de **bootstrap** qui contient les informations nécessaires pour restaurer précisement les fichiers que vous souhaitez. Les fichiers de **bootstrap** sont facultatifs mais fortement recommandés car c'est ce qui vous donne le plus de maîtrise sur quels fichiers doivent être restaurés. Pour plus de détails sur les fichiers de **bootstrap**, voyez le chapitre Restaurer des fichiers avec un fichier de Bootstrap de ce document. Il faut noter qu'un fichier de bootstrap peut être aussi généré par la commande **restore**. Par exemple :

```
./bextract -b bootstrap-file /dev/nst0 /tmp
```

Le fichier de bootstrap vous permet de décrire précisément quels fichiers doivent être restaurés (extraits). Vous pouvez utiliser un fichier de bootstrap en même temps que des fichiers d'inclusion et/ou d'exclusion. Les conditions du bootstrap seront appliquées en premier, et chaque enregistrement de fichier sera ensuite comparé par rapport aux listes d'inclusion et d'exclusion.

1.6.3 Extraire depuis plusieurs Volumes

Si vous souhaitez extraire des fichiers répartis sur plusieurs Volumes, vous pouvez indiquer les noms de Volumes dans le fichier de bootstrap ou bien en ligne commande, en les séparant par une barre verticale. Référez-vous à la section plus haut, appelée **Listing Multiple Volumes** (!!! FIXME!!!) pour plus de détails. La même méthode peut être utilisée avec le programme **bextract**, et vous pouvez également lire le chapitre Bootstrap de ce document.

1.7 bscan

Si vous devez utiliser ce programme, vous avez sans doute fait quelque chose de travers. Par exemple, la meilleurs méthode pour restaurer une base Bacula perdue ou endommagée est d'utiliser un fichier de bootstrap qui a été généré lors de la sauvegarde (comme par défaut dans le fichierbacula-dir.conf).

Le programme **bscan** est utilisé pour recréer les enregistrements de la base de données (catalogue) à partir des informations de sauvegarde contenues dans un ou plusieurs Volumes. Cela n'est normalement nécessaire que dans le cas où un ou plusieurs volumes ont été élagués ou purgés du catalogue (les enregistrements du volume ne sont alors plus dans le catalogue), ou pour les Volumes que vous avez archivés. Notez bien que si faites un scan de Volumes ayant été purgés auparavant, vous serez en mesure de faire des restaurations depuis ces volumes. Attention, à moins que vous ne changiez les délais de rétention au niveau Job et File pour les Jobs ajoutés par le scan, le prochain lancement d'un Job du même nom se traduira par nouvel élaguage des enregistrements. Etant donné que le scan des volumes est très long, ceci peut être très frustrant.

En faisant attention, **bscan** peut également être utilisé pour synchroniser un catalogue existant avec un Volume. Bien qu'il n'ait jamais été observé de cas où bscan a endommagé un catalogue, nous recommandons de faire un backup (dump) de votre base avant de lancer **bscan** pour être tranquille (car ce dernier modifie la base de données). Consultez Compacter Votre Base de Données pour les détails de la méthode pour faire une copie de votre base de données.

bscan peut également être utile dans une situation de récupération sur désastre, après la perte d'un disque dur si vous n'avez pas de fichier de **bootstrap** valide pour reinstaller votre système, ou bien si un Volume a été recyclé mais pas réécrit, vous pouvez utiliser **bscan** pour recréer votre base de données, qui peut ensuite être utilisée pour restaurer votre système ou un fichier dans son état précédent.

Il est appelé ainsi :

```
Usage: bscan [options] <bacula-archive>
       -b bootstrap   specify a bootstrap file
       -c <file>      specify configuration file
       -d <nn>        set debug level to nn
       -m             update media info in database
       -n <name>      specify the database name (default bacula)
       -u <user>      specify database user name (default bacula)
       -P <password>  specify database password (default none)
       -h <host>      specify database host (default NULL)
       -p             proceed inspite of I/O errors
       -r             list records
       -s             synchronize or store in database
       -v             verbose
       -V <Volumes>   specify Volume names (separated by |)
       -w <dir>       specify working directory (default from conf file)
       -?             print this message
```

Si vous utilisez MySQL ou PostgreSQL, il n'y a pas besoin d'indiquer un répertoire de travail (working directory) puisque dans ce cas, bscan sait où se trouvent les bases de données. Si vous avez protégé votre base de données vous devrez peut-être fournir soit le nom de la base (option **-b**), le nom d'utilisateur (option **-u**), et/ou le mot de passe (option **-p**).

NOTE : avant de pouvoir utiliser **bscan**, il vous faut au moins une base de données vide mais valide. Si votre base de données existe mais que certains enregistrements manquent, tout va bien. Si vous avez perdu ou détruit votre base de données, vous devez tout d'abord vous assurer que soit MySQL ou PostgresSQL fonctionnent, ensuite créer la base de données de Bacula (appelée bacula habituellement) et finalement créer les tables de Bacula en utilisant les scripts du répertoire **cats**. Ceci est expliqué au chapitre Installation de ce manuel. Pour terminer, avant de faire un scan vers une base de données vide, vous devez démarrer et arrêter le Director avec le von fichier bacula-dir.conf de sorte qu'il créee les enregistrements de Client et de Storage (qui ne sont pas stockés dans les Volumes). Sans ces enregistrements, le scan serait incapable de rapprocher les enregistrements de Job du bon Client.

Oublions pour l'instant la complexité d'une reconstruction complète de votre catalogue et supposons que vous ayez un backup dans les Volumes "Vol001" et "Vol002", et que par la suite tous les enregestriments d'un ou des deux Volumes sont élagués ou purgés de la base de données. En utilisant **bscan** vous pouver recréer les entrées du catalogue pour ces volumes et utiliser la commande **restore** dans la console pour restaurer ce que vous souhaitez. Une commande comme :

```
bscan -c bacula-sd.conf -v -V Vol001\|Vol002 /dev/nst0
```

vous donnera une idée de ce qui pourrait se passer sans changer votre catalogue. Vous aurez sans doute besoin de changer le chemin du fichier de configuration du Storage Daemon, le nom du Volume, et le nom de votre lecteur de bandes (ou de votre disque). Cette commande doit lire toute la bande, et si celle-ci contient beaucoup de données, cela prendra assez longtemps, ce qui fait que vous voudrez sans doute utiliser directement la commande indiquée ci-dessous. Notez bien que si vous sauvegardez vers un volume de disque, vous devez remplacer le nom de périphérique par le chemin du répertoire contenant les Volumes. Ceci doit correspondre à l'entrée "Archive Device" du fichier de configuration.

Pour ensuite stocker les enregistrements dans le catalogue, ajoutez l'option **-s** ainsi :

```
bcan -s -m -c bacula-sd.conf -v -V Vol001\|Vol002 /dev/nst0
```

Lors de l'écriture dans la base de donnnées, si bscan trouve des enregistrements existants, il va en général les mettre à jour s'ils sont mauvais, ou bien les laisser tels quels. Si les Volumes que vous scannez sont déjà

référencés dans le catalogue (en partie ou totalement), il ne sera fait aucun mal aux données existantes. Les données manquantes seront simplement ajoutées

Si vous avez plusieurs bandes, vous devez les scanner ainsi :

```
bscan -s -m -c bacula-sd.conf -v -V Vol001\|Vol002\|Vol003 /dev/nst0
```

Comme il y a une limite sur la longueur de la ligne de commande acceptée par **bscan** (511 octets), vous devrez créér manuellement un fichier de bootstrap si vous avez beaucoup de volumes. Référez-vous au chapitre Bootstrap de ce manuel pour plus de détails, en particulier la section appelée Bootstrap pour bscan.

Vous devriez toujours essayez de spécifier les bandes dans l'ordre dans lequel elles ont été écrites, même si bscan est capable de scanner des bandes non consécutives. Tout enregistrement incomplet à la fin de la bande sera tout simplement ignoré dans ce cas. Si vous réparez simplement un catalogue, cela peut suffire, mais si vous recréez un catalogue à partir de zéro, ceci laissera votre base de données dans un état inconsistant. Si vous ne spécifiez pas tous les volumes nécessaires en une seule commande, bscan ne sera pas capable de restaurer les enregistrements répartir sur deux volumes. En d'autres mots, il est préférable de spécifier deux ou trois volumes en une seule commande bscan plutôt que de lancer deux ou trois fois bscan, à chaque fois avec un seul volume.

Notez bien que le processus de restauration en utilisant bscan n'est pas identique à la création originale des données du catalogue. En effet, certaines données comme les enregistrements de Client ainsi que d'autres données non essentielles (comme les lectures de volumes, les montages de volumes, etc.) ne sont pas stockées dans le Volume, et donc non restaurées par bscan. Cependant, les résultats d'un bscan sont parfaitement valides et permettront de restaurer n'importe quel fichier du catalogue en utilisant les commandes habituelles de la console Bacula. Si vous partez d'un catalogue vide en espérant que bscan le reconstruira, vous serez un peu déçu, mais vous devez vous assurer au minimum que le fichier bacula-dir.conf est le même que précédemment – en clair il doit contenir toutes les bonnes ressources Client pour qu'elles soient recrées dans la nouvelle base de données **avant** de lancer bscan. Quand le Director démarre, il recrée tout enregistrement Client manquant du catalogue. Un autre problème pouvant subvenir est que même si les Volumes (enregistrements de Media) sont recrées dans la base de données, leur état d'autochanger et les slots ne seront pas correctement affectés. Vous devrez réparer ceci en utilisant la commande **update slots**. Vous pouvez également avoir d'autres problèmes. Avant de vous lancer dans un bscan, vous devez toujours essayer de remettre votre système en état à partir d'une sauvegarde du catalogue.

1.7.1 Utilisation de bscan pour comparer un Volume à un Catalogue existant

Si vous souhaitez comparer le contenu d'un Volume à un catalogue existant sans changer ce dernier, vous pouvez le faire en toute sécurité si et seulement si vous ne spécifiez **aucune** des deux options **-m** ou **-s**. Cependant, en l'état actuel du programme (Bacula version 1.26 !! FIXME !!), les fonctions de comparaison ne sont pas suffisamment approfondies, nous ne recommandons donc pas ce mode à d'autres fins que des tests

1.7.2 Utilisation de bscan pour recréer un Catalogue à partir d'un Volume

C'est dans ce mode que **bscan** est le plus utile. Vous pouvez soit faire un **bscan** vers un catalogue fraîchement créé, ou directement dans votre catalogue existant (après en avoir faire une copie de sauvegarde comme décrit plus haut. Vous devez normalement partir d'un catalogue fraîchement créé, ne contenant aucune donnée.

En partant d'un Volume seul, appelé **TestVolume1**, la commande est utilisée ainsi :

```
./bscan -V TestVolume1 -v -s -m -c bacula-sd.conf /dev/nst0
```

Dans le cas de volumes multiples, ajoutez-les au premier en les séparant par une barre verticale. Vous devrez peut-être préfixer la barre verticale d'un antislash pour l'échapper du shell – ie. **TestVolume1\|TestVolume2**. L'option **-v** a été ajoutée pour que le programme soit plus bavard. L'option **-s**

indique à **bscan** de stocker les données dans la base. Le nom du périphérique physique **/dev/nst0** est indiqué après toutes les options.

Par exemple, après avor fait une sauvegarde complète suivie de deux incrémentales d'un répertoire, j'ai réinitialisé la base de données SQLite comme décrit plus haut et en utilisant le fichier bootstrap.bsr indiqué plus haut, j'ai utilisé la commande suivante :

```
./bscan -b bootstrap.bsr -v -s -c bacula-sd.conf /dev/nst0
```

qui a généré l'affichage suivant :

```
bscan: bscan.c:182 Using Database: bacula, User: bacula
bscan: bscan.c:673 Created Pool record for Pool: Default
bscan: bscan.c:271 Pool type "Backup" is OK.
bscan: bscan.c:632 Created Media record for Volume: TestVolume1
bscan: bscan.c:298 Media type "DDS-4" is OK.
bscan: bscan.c:307 VOL_LABEL: OK for Volume: TestVolume1
bscan: bscan.c:693 Created Client record for Client: Rufus
bscan: bscan.c:769 Created new JobId=1 record for original JobId=2
bscan: bscan.c:717 Created FileSet record "Kerns Files"
bscan: bscan.c:819 Updated Job termination record for new JobId=1
bscan: bscan.c:905 Created JobMedia record JobId 1, MediaId 1
bscan: Got EOF on device /dev/nst0
bscan: bscan.c:693 Created Client record for Client: Rufus
bscan: bscan.c:769 Created new JobId=2 record for original JobId=3
bscan: bscan.c:708 Fileset "Kerns Files" already exists.
bscan: bscan.c:819 Updated Job termination record for new JobId=2
bscan: bscan.c:905 Created JobMedia record JobId 2, MediaId 1
bscan: Got EOF on device /dev/nst0
bscan: bscan.c:693 Created Client record for Client: Rufus
bscan: bscan.c:769 Created new JobId=3 record for original JobId=4
bscan: bscan.c:708 Fileset "Kerns Files" already exists.
bscan: bscan.c:819 Updated Job termination record for new JobId=3
bscan: bscan.c:905 Created JobMedia record JobId 3, MediaId 1
bscan: Got EOF on device /dev/nst0
bscan: bscan.c:652 Updated Media record at end of Volume: TestVolume1
bscan: bscan.c:428 End of Volume. VolFiles=3 VolBlocks=57 VolBytes=10,027,437
```

Il faut bien noter ici que **bscan** affiche une ligne à chaque enregistrement majeur qu'il créé. Vu le grand volume d'affichage potentiel, il n'affiche pas une ligne pour chaque enregistrement de fichier, à moins qu'il ne trouve l'option **-v** au moins deux fois sur la ligne de commande.

Dans le cas d'un enregistrement de Job, le nouveau JobId ne sera pas le même que le JobId initial. Par exemple, pour le premier JobId rencontré ci-dessus, le nouveau JobId est 1 alors que le JobId initial était 2. Il n'y a pas de souci à se faire, c'est le fonctionnement normal des bases de données. **bscan** se chargera de tout conserver de manière homogène.

Bien que **bscan** dise qu'il a créé un enregistrement de Client pour le Client : Rufus trois fois, il ne le fait effectivement que la première fois. C'est le comportement attendu.

Vous avez peut-être remarqué que bscan a rencontré une marque de fin de fichier après chaque Job (Got EOF on device ...). Pour terminer, la dernière ligne contient les statistiques du bscan.

Si vous aviez ajouté une seconde option **-v** à la ligne de commande, Bacula aurait été encore plus bavard, en affichant pratiquement tous les détails de chaque enregistrement de Job rencontré.

Maintenant si vous démarrez Bacula et tapez la commande **list jobs** dans la console, vous obtenez :

```
+-------+----------+-----------------+------+-----+----------+----------+---------+
| JobId | Name     | StartTime       | Type | Lvl | JobFiles | JobBytes | JobStat |
+-------+----------+-----------------+------+-----+----------+----------+---------+
| 1     | kernsave | 2002-10-07 14:59 | B    | F   | 84       | 4180207  | T       |
| 2     | kernsave | 2002-10-07 15:00 | B    | I   | 15       | 2170314  | T       |
| 3     | kernsave | 2002-10-07 15:01 | B    | I   | 33       | 3662184  | T       |
+-------+----------+-----------------+------+-----+----------+----------+---------+
```

ce qui correspond pratiquement à l'identique à la base de données avant qu'elle soit réinitialisée et restaurée avec bscan. Tous les Jobs et Files trouvés sur la bande ont été restaurés, y compris la plupart des enregistrements de Media. Les enregistrements de Volume (Media) restaurés sont marqués comme **Full** pour qu'ils ne puissent pas être écrasés sans intervention de l'opérateur.

Il faut bien voir que **bscan** ne peut pas restaurer une base de données dans son état exact précédent car une grande partie des informations moins importantes ne sont pas stockées sur la bande. Malgré tout, la reconstruction est assez complète pour utiliser la commande **restore** et obtenir des résultats valides.

Un aspect intéresant d'une restauration de catalogue avec **bscan** est le cas où la sauvegarde est faite pendant que Bacula fonctionne et est en train d'écrire sur la bande. Au moment où la sauvegarde du catalogue est faite, la bande que Bacula écrit contiendra, disons, 10 fichiers, mais après la sauvegarde du catalogue, il y aura 11 fichiers sur la bande. Il y a donc une différence entre le contenu du catalogue et celui de la bande. Si après une restauration du catalogue vous tentez d'écrire sur la même bande (celle utilisée pour sauvegarder le catalogue), Bacula détectera une différence entre le nombre de fichiers enregistrés dans le catalogue et ce qu'il y a sur la bande, et marquera cette dernière en erreur.

Il y a deux solutions à ce prolbème. La première et sans doute la plus simple est de marquer le volume comme Used avant toute sauvegarde. La seconde est de corriger à la main le nombre de fichiers associé à l'enregistrement de Media du catalogue. Cette procédure est documentée ailleurs dans le manuel et implique d'utiliser la commande **update volume** dans la **bconsole**.

1.7.3 Utilisation de bscan pour corriger le nombre de fichiers d'un Volume

Si le Storage daemon plante pendant un Job de sauvegarde, le catalogue ne sera pas mis à jour correctement pour le volume en cours d'utilisation au moment du plantage. Ceci signifie que le Storage daemon aura écrit, disons 20 fichiers sur la bande, mais l'enregistrement de Volume du catalogue n'indiquera que 19 fichiers.

Bacula refuse d'écrire sur une bande qui contient un nombre différent de fichiers par rapport au catalogue. Pour corriger ce problème, vous pouvez lancer un **bscan** avec l'option **-m** (mais **sans** l'option **-s**) pour mettre à jour seulement l'enregistrement de Media pour les Volumes lus.

1.7.4 Après bscan

Si vous utilisez **bscan** pour entrer le contenu d'un Volume dans un catalogue existant, vous devez être conscient que les enregistrements ajoutés peuvent être immédiatement élagués lors du prochain job, surtout si le Volume est très ancien ou a été purgé précédemment. Pour éviter ceci, vous pouvez, après avoir lancé **bscan**, positionner l'état du Volume dans le catalogue (VolStatus) à la valeur **Read-Only** en utilisant la commande **update**. Ceci vous permettra de restaurer depuis ce volume sans risquer qu'il soit immédiatement purgé. Uns fois vos données restaurées et à l'abri, vous pouvez réinitialiser le VolStatus à **Used** et le Volume sera purgé du catalogue.

1.8 bcopy

Le programme **bcopy** peut être utilisé pour copier une archive **Bacula** vers une autre. Vous pouvez par exemple copier une bande vers un fichier, un fichier vers une bande, un fichier vers un fichier ou une bande vers une bande. Pour du bande vers bande, vous devez disposer de deux lecteurs de bande. (une version future est planifiée, avec la possibilité de se servir du disque comme tampon). Le processus de copie n'écrit aucune information dans le catalogue à propos du nouveau Volume. Ceci signifie que ce nouveau Volume, même s'il contient des données de sauvegarde valides, ne peut pas être accédé directement à partir des entrées du catalogue. Si vous souhaitez pouvoir utiliser le Volume avec la commande restore, vous devez d'abord le passer à bscan pour l'intégrer au catalogue.

1.8.1 Options de la commande bcopy

```
Usage: bcopy [-d debug_level] <input-archive> <output-archive>
       -b bootstrap      specify a bootstrap file
       -c <file>         specify configuration file
       -dnn              set debug level to nn
       -i                specify input Volume names (separated by |)
       -o                specify output Volume names (separated by |)
       -p                proceed inspite of I/O errors
       -v                verbose
       -w dir            specify working directory (default /tmp)
       -?                print this message
```

En utilisant un fichier de **bootstrap**, vous pouvez copier une partie du contenu d'une archive Bacula vers une autre archive.

Un des objectifs de programme est de récupérer autant de données que possible d'une bande endommagée. Cependant, cette version ne dispose pas encore de cette fonctionnalité.

!!!! FIXME!!!! Ce programme étant assez récent, tout retour d'utilisation sera apprécié. Je ne dispose de plus que d'un seul lecteur de bandes, je n'ai donc pas testé ce programme avec deux lecteurs.

1.9 btape

Ce programme permet d'effectuer un bon nombre d'opérations élémentaires sur une bande à partir d'un simple interface terminal. Il ne fonctionne qu'avec les bandes, et non pas avec les autres supports autorisés par Bacula (DVD, File, etc.). La commande **test**, décrite plus bas, peut être très utile pour diagnostiquer les problèmes de compatibilité de lecteurs un peu anciens. A part le test initial de compatibilit des lecteurs de bande avec **Bacula**, **btape** sera principalement utilisé par les développeurs pour écrire de nouveaux pilotes de lecteurs de bande.

btape peut être dangereux à utiliser avec des bandes **Bacula** existantes car il va re-étiqueter une bande ou bien écrire des données dessus sans se soucier de la présence de données précieuses. Faites très attention, et ne l'utilisez que sur des bandes vierges.

Pour fonctionner correctement, **btape** a besoin de lire le fichier de configuration du Storage daemon. Par défaut, il cherchera un fichier nommé **bacula-sd.conf** dans le répertoire courant. Si votre fichier de configuration se trouve ailleurs, utilisez l'option **-c** pour lui indiquer où.

Le nom du périphérique physique doit être indiqué en ligne de commande, et ce même nom doit être présent dans le fichier de configuration du Storage daemon lu par **btape**.

```
Usage: btape <options> <device_name>
       -b <file>   specify bootstrap file
       -c <file>   set configuration file to file
       -d <nn>     set debug level to nn
       -p          proceed inspite of I/O errors
       -s          turn off signals
       -v          be verbose
       -?          print this message.
```

1.9.1 Utilisation de btape pour vérifier votre lecteur de bandes

Une des raisons d'être de ce programme est de s'assurer que le fichier de configuration du Storage Daemon est correctement défini pour que Bacula puisse lire et écrire correctement des bandes.

Il est fortement recommandé d'utiliser la commande **test** avant de lancer votre premier job Bacula pour être bien sûr que les paramètres du périphérique de stockage (lecteur de bande) permettront à **Bacula** de fonctionner correctement. Vous devez simplement monter une bande vierge, lancer la commande et l'affichage

sera assez clair pour se passer d'explications. Référez-vous au chapitre Test des bandes de ce manuel pour les détails.

1.9.2 Commandes de btape

Voilà la liste complète des commandes :

```
Command    Description
=======    ===========
autochanger test autochanger
bsf        backspace file
bsr        backspace record
cap        list device capabilities
clear      clear tape errors
eod        go to end of Bacula data for append
eom        go to the physical end of medium
fill       fill tape, write onto second volume
unfill     read filled tape
fsf        forward space a file
fsr        forward space a record
help       print this command
label      write a Bacula label to the tape
load       load a tape
quit       quit btape
rawfill    use write() to fill tape
readlabel  read and print the Bacula tape label
rectest    test record handling functions
rewind     rewind the tape
scan       read() tape block by block to EOT and report
scanblocks Bacula read block by block to EOT and report
speed      report drive speed
status     print tape status
test       General test Bacula tape functions
weof       write an EOF on the tape
wr         write a single Bacula block
rr         read a single record
qfill      quick fill command
```

Les commandes les plus utiles sont :

- `test` – test d'écriture d'enregistrements et de marques de fin de fichier (EOF), puis test de lecture.
- `fill` – remplissage complet d'un volume avec des enregistrements, puis écriture de quelques enregistrements dans un second volume. Pour terminer, les deux volumes sont relus. Cette commande écrit des blocs de données aléatoires, pour que votre lecteur de bandes ne puisse pas compresser les données. C'est donc un bon test pour estimer la contenance physique réelle de vos bandes.
- `readlabel` – lit et affiche l'étiquette (label) d'une bande Bacula.
- `cap` – affiche les fonctionnalités du périphérique définies dans le fichier de configuration et comment elles sont perçues par le Storage Daemon.

La commande **readlabel** peut être utilisée pour afficher les détails d'une étiquette de bande Bacula. Ceci peut être utile si l'étiquette de la bande est pardue ou endommagée.

Au cas où vous souhaitiez re-étiquer une bande **Bacula**, vous pouvez simplement utiliser la commande **label** qui écrasera l'ancienne étiquette. Cependant, nous recommandons d'utiliser la commande **label** de la **Console** car elle n'écrasera jamais une bande Bacula valide.

Tester votre lecteur de bandes

Pour déterminer la meilleure configuration de votre lecteur de bandes, vous pouvez utiliser la commande `speed` du programme `btape`.

La commande peut accepter les arguments suivants :
`file_size=n` indique le Maximum File Size pour ce test (entre 1 et 5GB). L'unité est le GB.

`nb_file=n` indiquer le nombre de fichiers à écrire. La taille totale des données écrites devraient être supérieure à la quantité de mémoire de la machine ($file\_size * nb\_file$).
`skip_zero` ce drapeau permet de passer les tests de données constantes.
`skip_random` ce drapeau permet de passer les tests de données aléatoires.
`skip_raw` ce drapeau permet de passer les tests en accès direct.
`skip_block` ce drapeau permet de passer les tests en accès par bloc Bacula.

```
*speed file_size=3 skip_raw
btape.c:1078 Test with zero data and bacula block structure.
btape.c:956 Begin writing 3 files of 3.221 GB with blocks of 129024 bytes.
++++++++++++++++++++++++++++++++++++++++
btape.c:604 Wrote 1 EOF to "Drive-0" (/dev/nst0)
btape.c:406 Volume bytes=3.221 GB. Write rate = 44.128 MB/s
...
btape.c:383 Total Volume bytes=9.664 GB. Total Write rate = 43.531 MB/s

btape.c:1090 Test with random data, should give the minimum throughput.
btape.c:956 Begin writing 3 files of 3.221 GB with blocks of 129024 bytes.
++++++++++++++++++++++++++++++++++++++++
btape.c:604 Wrote 1 EOF to "Drive-0" (/dev/nst0)
btape.c:406 Volume bytes=3.221 GB. Write rate = 7.271 MB/s
++++++++++++++++++++++++++++++++++++++++
...
btape.c:383 Total Volume bytes=9.664 GB. Total Write rate = 7.365 MB/s
```

Dans le cas de la compression matérielle, le test aléatoire vous donnera le début minimum de votre lecteur. Le test à données constantes vous donnera la débit maximum de votre chaîne matérielle. (processeur, mémoire, carte SCSI, câble, lecteur, bande)

Vous pouvez changer la taille des blocs dans le fichier de configuration du Storage Daemon

1.10 Autres Programmes

Les programmes suivants n'ont pas forcément besoin de fichier de configuration ou de nom de périphérique.

1.11 bsmtp

bsmtp est un simple programme MTA (Mail Transfer Agent) qui apporte un peu plus de souplesse que les programmes que l'on trouve généralement sur les systèmes UNIX. Il peut même être utilisé sur des machines Windows.

Il est appelé de la façon suivante :

```
Usage: bsmtp [-f from] [-h mailhost] [-s subject] [-c copy] [recipient ...]
       -c          set the Cc: field
       -dnn        set debug level to nn
       -f          set the From: field
       -h          use mailhost:port as the bsmtp server
       -l          limit the lines accepted to nn
       -s          set the Subject: field
       -?          print this message.
```

Si l'option **-f** n'est pas spécifiée, **bsmtp** utilisera votre identifiant (userid). Si l'option **-h** n'est pas spécifiée, **bsmtp** utilisera la variable d'environnement **bsmtpSERVER** ou bien **localhost** si **bsmtpSERVER** n'existe pas. Par défaut le port 25 est utilisé.

Si une limite du nombre de lignes est indiquée avec l'option **-l**, **bsmtp** n'enverra pas de mail avec un corps de message dépasant ce nombre de lignes. C'est très utile pour les rapports de gros jobs de restauration où a liste des fichiers restaurés peut produire des mails très longs que votre serveur de mail pourrait refuser (ou qui pourraient le planter). Cependant, il faut être conscient du fait que vous risquez de rater le rapport du job et tout message d'erreur rencontré, à moins de consulter le fichier de log produit par le Director (pour plus de détails, voir la ressource **Messages** de ce manuel)

L'argument **recipients** est la liste des destinataires du mail, séparés par des espaces.

Le corps du message est lu sur l'entrée standard.

Un exemple d'utilisation de **bsmtp** serait de mettre les directives suivantes dans la ressource **Messages** de votre fichier **bacula-dir.conf**. Attention, ces commandes doivent être saisies sur une seule ligne chacune.

```
mailcommand = "/home/bacula/bin/bsmtp -h mail.domain.com -f \"\(Bacula\) %r\"
              -s \"Bacula: %t %e of %c %l\" %r"
operatorcommand = "/home/bacula/bin/bsmtp -h mail.domain.com -f \"\(Bacula\) %r\"
                  -s \"Bacula: Intervention needed for %j\" %r"
```

Où vous remplacez **/home/bacula/bin** par le chemin d'accès à vos exécutables de **Bacula** et **mail.domain.com** par le nom d'hôte complètement qualifié de votre serveur SMTP, qui écoute normalement sur le port 25. Pour plus de détails sur les caractères de substitution (ie. %r) utilisés ci-dessus, référez-vous au chapitre MailCommand de la ressource Messages de ce manuel.

Il est TRES fortement recommandé de tester manuellement un ou deux cas pour être sur que le **mailhost** spécifié est correct et qu'il accepte bien vos envois. Comme **bsmtp** utiliser toujours une connection TCP plutôt que d'écrire dans le spool, vous constaterez peut-être que votre adresse **from** est rejetée car elle ne contient pas de domaine valide, ou bien parce que votre message déclenche une règle de filtrage antispam. En général, il est conseillé d'utiliser un nom de domaine complètement qualifié dans le champ **from**, et suivant si votre passerelle SMTP est Exim ou Sendmail, vous devrez peut-être modifier la syntaxe du "From :" du message. N'hésitez pas à tester.

En utilisant **bsmtp** à la main, vous devrez terminer votre message par un CTRL-D sur la colonne 1 de la dernière ligne.

Si vous obtenez des dates invalides (comme 1970) et que vous êtes en environnement de langue autre que l'Anglais, vous pouvez essayer d'ajouter un `LANG="en_US"` juste avant l'appel de bsmtp.

bsmtp tente généralement d'assainir le contenu des champs from, copy, mailhost, et recipient, en ajoutant les caractères < et > nécessaires autour de la partie adresse. Cependant, si vous indiquez un **display-name** (voir RFC 5332), certains serveurs SMTP comme Exchange pourraient ne pas accepter le message si le **display-name** est présent entre les caractères < et >. Comme il est dit plus haut, vous devez tester, et si vous rencontrez ce problème, vous pouvez ajouter à la main les caractères < et > aux commandes **mailcommand** ou **operatorcommand** : quand **bsmtp** remettra l'adresse en forme, il la laissera inchangée s'il trouve déjà les caractères < ou >.

1.12 dbcheck

dbcheck est un programme qui rechercha des incohérences logiques dans les tables de votre base de données Bacula, et éventuellement les corrigera. Ce programme est une routine de maintenance de la base de données, dans le sens où il détecte et supprime les lignes inutilisées, mais ce n'est pas un outil de réparation de base de données. Pour réparer une base de données, il faut utiliser les outils fournis avec votre système de base de données. dbcheck n'a normalement pas besoin d'être utilisé, sauf dans les cas de crash de Bacula ou bien si vous avez beaucoup de Clients, de Pools ou de Jobs ayant été supprimés.

Le programme **dbcheck** est disponible dans le répertoire <**bacula-source**>**/src/tools** des sources de Bacula. Bien qu'il soit compilé lors du make, il n'est normalement pas "installé".

Il est appelé ainsi :

```
Usage: dbcheck [-c config ] [-B] [-C catalog name] [-d debug_level]
  <working-directory> <bacula-database> <user> <password> [<dbhost>] [<dbport>]
       -b              batch mode
       -C              catalog name in the director conf file
       -c              Director conf filename
       -B              print catalog configuration and exit
       -d <nn>         set debug level to <nn>
       -dt             print timestamp in debug output
       -f              fix inconsistencies
       -v              verbose
       -?              print this message
```

Si l'option **-B** est spécifiée, dbcheck affichera les informations de connexion au catalogue en texte brut. C'est utile pour le sauvegarde de manière sécurisée.

```
$ dbcheck -B
catalog=MyCatalog
db_type=SQLite
db_name=regress
db_driver=
db_user=regress
db_password=
db_address=
db_port=0
db_socket=
```

Si l'option **-c** est donnée avec le fichier de configuration du Director, il n'y a besoin d'aucun autre argument, en particulier le répertoire de travail car dbcheck le lira dans le fichier.

Si l'option **-f** est spécifiée, **dbcheck** réparera (**fix**) les incohérences qu'il trouvera. Dans le cas contraire, il sera contentera de les rapporter.

Si l'option **-b** est spécifiée, **dbcheck** fonctionnera en mode batch, il lancera tout de suite l'examen et la correction (si l'option -f est présente) de toutes les incohérences. Si l'option **-b** n'est pas spécifiée, **dbcheck** sera lancé en mode interactif et affichera un menu tel que :

```
Hello, this is the database check/correct program.
Please select the function you want to perform.
     1) Toggle modify database flag
     2) Toggle verbose flag
     3) Repair bad Filename records
     4) Repair bad Path records
     5) Eliminate duplicate Filename records
     6) Eliminate duplicate Path records
     7) Eliminate orphaned Jobmedia records
     8) Eliminate orphaned File records
     9) Eliminate orphaned Path records
    10) Eliminate orphaned Filename records
    11) Eliminate orphaned FileSet records
    12) Eliminate orphaned Client records
    13) Eliminate orphaned Job records
    14) Eliminate all Admin records
    15) Eliminate all Restore records
    16) All (3-15)
    17) Quit
Select function number:
```

En entrant 1 ou 2, vous pouvez bascule d'un état à l'autre pour les drapeaux de modification de la base de données (option -f) et pour le drapeau du mode bavard (-v). Il est utile et rassurant de désactiver la modification de la base de données, de lancer un ou plusieurs tests de cohérence (options 3 à 9) pour voir ce qui serait fait, puis de se mettre en mode modification, et de nouveau lancer les tests.

Les incohérences détectées sont les suivantes :

- Duplicate filename records (Doublons dans les noms de fichiers). Ceci peut se produire si deux instances de Bacula sont exécutées simultanément, et que les deux ajoutent des noms de fichiers en même temps. Ceci arrive rarement mais la base de données se retrouve alors dans un état incohérent. Dans ce cas, vous rencontrerez des messages d'erreur pendant les Jobs qui avertiront de doublons dans la base de données. Si vous ne rencontrez pas ces erreurs, vous n'avez aucune raison de lancer cette vérification.
- Repair bad Filename records (Réparation des noms de fichiers erronés) . Ce test vérifie et corrige les noms de fichiers se terminant par un slash (ils ne doivent pas en avoir)
- Repair bad Filename records. This checks and corrects filenames that have a trailing slash. They should not.
- Repair bad Path records (Réparation des chemins erronés). Ce test vérifie et corrige les noms de chemins qui ne se terminent pas par un slash (ils doivent en avoir).
- Duplicate path records (Doublons dans les noms de chemins). Ceci peut se produire si deux instances de Bacula sont exécutées simultanément, et que les deux ajoutent des noms de fichiers en même temps. Ceci arrive rarement mais la base de données se retrouve alors dans un état incohérent. Voyez plus haut pour les explications.
- Orphaned JobMedia records (JobMedia orphelins). Ceci se produit quand un enregistrement de Job est supprimé (directement en SQL par l'utilisateur par exemple), mais que le JobMedia correspondant (un pour chaque Volume utilisé par le Job) n'est pas supprimé. Ceci ne doit normalement pas se produire, et même si cela arrive, ces enregistrements n'occupent pas beaucoup d'espace dans la base de données. Si vous le souhaitez, vous pouver utiliser cette vérification pour supprimer ces enregistrements orphelins.
- Orphaned File records (Fichiers orphelins). Ceci se produit quand un enregistrement de Job est supprimé (directement en SQL par l'utilisateur par exemple), mais que les enregistrements de Fichiers correspondants (un pour chaque Volume utilisé par le Job) ne sont pas supprimés. Attention, la recherche ce des enregistrements peut être **très** longue (en heures) sur une base de données importantes. Ceci ne doit normalement pas se produire car Bacula fait tout pour l'empêcher. Il est cependant recommandé de lancer cette vérification une fois par an car les Fichiers orphelins peuvent occuper beaucoup d'espace dans votre base de données. Assurez-vous d'avoir des index sur les champs JobId, FilenameId, et PathId de la table File de votre catalogue avant de lancer cette commande.
- Orphaned Path records (Chemins orphelins). Ceci se produit à chaque fois qu'un répertoire est supprimé du système et tous que les enregistrements de Job associés ont été purgés. Lors de la purge ou de l'élaguage des Jobs, Bacula ne vérifie pas la présence de chemins orphelins. Avec le temps, les enregistrements de chemins inutilisés s'accumulent et occupent de l'espace dans votre base de données. Cette vérification les supprimera et il est conseillé de la faire une fois par an.
- Orphaned Filename records (Noms de fichiers orphelins). Ceci se produit à chaque fois qu'un fichier est supprimé du système et que tous les enregistrements de Job associés ont été purgés. Ceci peut se produire fréquemment quand beaucoup de fichiers sont créés puis supprimés. De plus, si vous faites une mise à jour système ou si vous supprimez tous une arborescence, il peut rester beaucoup d'enregistrements de noms de fichiers inutilisés dans le catalogue.
  Lors de la purge (ou de l'élaguage) des Jobs, Bacula ne vérifie pas la présence de noms de fichiers orphelins. Avec le temps, les enregistrements de noms de fichier inutilisés s'accumulent et occupent de l'espace dans votre base de données. Cette vérification les supprimera, et il est fortement recommandé de la lancer au moins une fois par an. Il est sans doute préférable de le faire tous les 6 mois dans le cas des groses bases de données (200 Mo et plus)
- Orphaned Client records (Clients orphelins). Ces enregistrements peuvent rester dans la base de données après la suppression d'un client.
- Orphaned Job records (Jobs orphelins). Si aucun client n'est associé à un job, ou bien qu'un job n'est pas utilisé pendant longtemps, ces enregistrements peuvent s'accumuler. Cette option vous permet de supprimer les jobs qui ne sont associés à aucune client (et donc inutiles)
- All Admin records (tous les enregistrements "Admin"). Cette commande supprimera tous les enregistrements "Admin", quel que soit leur âge.
- All Restore records (tous les enregistrements "Restore"). Cette commande supprimera tous les enregistrements "Restore", quel que soit leur âge.

Si vous utilisez MySQL, dbcheck vous demandera si vous souhaitez créer des index temporaires pour accélérer la suppression des chemins et noms de fichier orphelins

Principalement à destination des utilisateurs de PostgreSQL, nous fournissons un remplacement à dbcheck sous forme d'un script SQL pur, disponible dans in `examples/database/dbcheck.sql` et qui fonctionne avec des requêtes globales au lieu de plusieurs petites requêtes comme dbcheck le fait. Vous trouverez des instructions au début du script et vous devrez taper la commande `COMMIT` à la fin pour confirmer les modifications.

Si vous utilisez bweb ou brestore, ne supprimez pas les chemins orphelins ou bien vous devrez reconstruire les index `brestore_pathvisibility` et `brestore_pathhierarchy`.

Pour finir, je n'utilise personnellement dbcheck que lorsque j'ai planté ma base de données à cause d'un bug lors du développement de Bacula, ce qui fait que vous ne devriez jamais avoir besoin d'utiliser dbcheck malgré les recommandations ci-dessus, qui sont données pour éviter que des utilisateurs ne perdent trop de temps en se servant de dbcheck trop souvent.

1.13 bregex

bregex est un petit programme qui vous permettra de tester des expressions regulières sur des fichiers ou des données. Ceci peut être utile car les bibliothèques d'expressions régulières diffèrent suivant le système, ce qui ajoute encore à leur complexité inhérente.

bregex se trouve dans le répertoire `src/tools` et est installé normalement avec les autres exécutables système. Vous pouvez l'utiliser ainsi :

```
Usage: bregex [-d debug_level] -f <data-file>
       -f          specify file of data to be matched
       -l          suppress line numbers
       -n          print lines that do not match
       -?          print this message.
```

L'argument <`data-file`> correspond au nom d'un fichier contenant les données (lignes) devant être appariés (ou non) par rapport à un ou plusieurs motifs. Quand le programme est exécuté, il vous demande un motif d'expression régulière, et l'applique à chaque ligne du fichier. Toute ligne qui correspond sera affichée, précédée du numéro de la ligne. Le programme vous demandera un nouveau motif pour continuer si vous le souhaitez.

Il suffit de saisir une ligne de vide pour que le programme se termine. Vous pouvez choisir de n'afficher que lignes ne correspondant pas avec l'option -n, et vous pouvez empêcher l'affichage des numéros de ligne avec l'option -l.

Ce programme peut être utile pour tester des expressions regulières qui doivent être appliquées sur une liste de noms de fichiers.

1.14 bwild

bwild est un petit programme qui vous permettra de tester des expressions à base de caractèrs joker (wild-cards) sur le contenu d'un fichier.

bwild se trouve dans le répertoire `src/tools` et est installé normalement avec les autres exécutables système. Vous pouvez l'utiliser ainsi :

```
Usage: bwild [-d debug_level] -f <data-file>
       -f          specify file of data to be matched
       -l          suppress line numbers
       -n          print lines that do not match
       -?          print this message.
```

L'argument <`data-file`> correspond au nom d'un fichier contenant les données (lignes) devant être appariés (ou non) par rapport à un ou plusieurs motifs. Quand le programme est exécuté, il vous demande un motif de jokers, et l'applique à chaque ligne du fichier. Toute ligne qui correspond sera affichée, précédée du numéro de la ligne. Le programme vous demandera un nouveau motif pour continuer si vous le souhaitez.

Il suffit de saisir une ligne de vide pour que le programme se termine. Vous pouvez choisir de n'afficher que lignes ne correspondant pas avec l'option -n, et vous pouvez empêcher l'affichage des numéros de ligne avec l'option -l.

Ce programme peut être utile pour tester des expressions à jokers qui doivent être appliquées sur une liste de noms de fichiers.

1.15 testfind

testfind permet de lister des fichiers avec le même moteur de recherche que celui utilisé par la ressource **Include** d'un Job. Il faut noter qu'une grande partie des fonctionnalités de ce programme (liste des fichiers à inclure) est présente dans la commande estimate de la Console.

L'utilité initiale de testfind était de s'assurer sur le moteur de recherche de fichiers de Bacula fonctionnait correctement, et d'afficher quelques statistiques sur les noms de fichiers et leur longueur. Cependant, vous pouvez trouver un utilité à ce programme pour voir ce que Bacula ferait avec une ressource **Include** donnée. Le programme **testfind** se trouve dans le répertoire <**bacula-source**>**/src/tools** des sources de Bacula. Bien qu'il soit compilé lors du make, il n'est normalement pas "installé".

Le programme est appelé ainsi :

```
Usage: testfind [-d debug_level] [-] [pattern1 ...]
       -a          print extended attributes (Win32 debug)
       -dnn        set debug level to nn
       -           read pattern(s) from stdin
       -?          print this message.
Patterns are used for file inclusion -- normally directories.
Debug level>= 1 prints each file found.
Debug level>= 10 prints path/file for catalog.
Errors are always printed.
Files/paths truncated is a number with len> 255.
Truncation is only in the catalog.
```

Où un "pattern" est n'importe quelle spécification de nom de fichier valide dans une définition de ressource **Include**. Si aucun "pattern" n'est spécifié, / (le répertoire racine) est utilisé. Par exemple :

```
./testfind /bin
```

produira l'affichage suivant :

```
Dir: /bin
Reg: /bin/bash
Lnk: /bin/bash2 -> bash
Lnk: /bin/sh -> bash
Reg: /bin/cpio
Reg: /bin/ed
Lnk: /bin/red -> ed
Reg: /bin/chgrp
...
Reg: /bin/ipcalc
Reg: /bin/usleep
Reg: /bin/aumix-minimal
Reg: /bin/mt
Lnka: /bin/gawk-3.1.0 -> /bin/gawk
Reg: /bin/pgawk
Total files    : 85
Max file length: 13
Max path length: 5
Files truncated: 0
Paths truncated: 0
```

Même si **testfind** utilise le même moteur de recherche que **Bacula**, chaque répertoire à lister doit être fourni séparement en ligne de commande ou un par ligne sur l'entrée standard, même si l'option **-** est spécifiée

Un niveau de debug de 1 (i.e. **-d1**) sur la ligne de commande poussera **testfind** à afficher les noms de fichiers bruts sans montrer les types de fichiers internes de Bacula, où le lien (s'il existe). Les niveaux de debug supérieurs ou égaux à 10 déclenchent l'affichage du nom du fichier et du chemin séparément en utilisant le même algorithme que celui utilisé pour stocker les noms de fichier dans la base de donnnées du Catalogue

b

1.16 bimagemgr

bimagemgr est un utilitaire destiné à ceux qui sauvegardent vers des volumes disque afin de les graver ensuite sur CD, plutôt que vers des bandes. C'est une interface web écrite en Perl qui est utilisée pour détecter quand un volume doit être gravé. Il nécessite :

- Un serveur web fonctionnant sur le serveur Bacula
- Un graveur de CD installé et configuré sur le serveur Bacula
- Le paquet cdrtools installé sur le serveur Bacula
- perl, le module perl-DBI, et l'un des modules DBD-MySQL, DBD-SQLite ou DBD-PostgreSQL.

Le gravage de DVD n'est pas supporté actuellement par **bimagemgr** mais c'est prévu pour une future version.

1.16.1 Installation de bimagemgr

Installation depuis les sources :

1. Vérifiez le Makefile et adaptez-le à votre configuration au besoin.
2. Adaptez config.pm à votre configuration.
3. Lancez un 'make install' en tant que root.
4. Modifiez votre `httpd.conf` et modifiez la valeur de la directive Timeout value. Le serveur web ne doit pas expirer et donc couper la connexion avant que le gravage ne soit terminé. La valeur exacte dépend de la vitesse de votre graveur et du fait que vous graviez le CD sur le serveur Bacula ou bien sur une autre machine en réseau. Dans mon cas, je l'ai positionné à 1000 secondes. Redémarrez httpd par la suite pour prendre en compte vos modifications.
5. Assurez-vous que cdrecord est bien setuid root.

Installation depuis des paquets RPM :

1. Installez le paquet RPM correspondant à votre plate-forme.
2. Modifiez /cgi-bin/config.pm pour l'adapter à votre configuration.
3. Modifiez votre `httpd.conf` et modifiez la valeur de la directive Timeout value. Le serveur web ne doit pas expirer et donc couper la connexion avant que le gravage ne soit terminé. La valeur exacte dépend de la vitesse de votre graveur et du fait que vous graviez le CD sur le serveur Bacula ou bien sur une autre machine en réseau. Dans mon cas, je l'ai positionné à 1000 secondes. Redémarrez httpd par la suite pour prendre en compte vos modifications.
4. Assurez-vous que cdrecord est bien setuid root.

Pour les versions de Bacula inférieures à la 1.36 :

1. Adaptez la partie configurable de config.pm à votre environnement.
2. Lancez `/etc/bacula/create_cdimage_table.pl` depuis une console en root de votre serveur Bacula pour ajouter la table CDImage à votre base de données Bacula.

Accéder aux fichiers de Volume : Les fichiers de Volume ont des permissions à 640 par défaut, et ne peuvent donc être lues que par root. L'approche recommandée est la suivante (ceci fonctionne seulement si bimagemgr et Apache fonctionnent sur le même serveur que Bacula) :

Pour Bacula en 1.34 ou 1.36 installé depuis les sources :

1. Ajoutez un nouveau groupe d'utilisateurs appelé "bacula" et ajoutez l'utilisateur "apache" à ce groupe (Redhat/Mandrake). Sur une SuSE, l'utilisateur à ajouter au groupe est "wwwrun".
2. Modifiez le propriétaire des tous les fichiers de Volume à root.bacula
3. Modifiez le script de démarrage `/etc/bacula/bacula` pour positionner `SD_USER=root` et `SD_GROUP=bacula`. Redémarrez Bacula.

Note : L'étape 3 devrait aussi comprendre la modification de `/etc/init.d/bacula-sd` mais les versions de ce fichier antérieures à la 1.36 ne le supportent pas. Il serait nécessaire dans ce cas de lancer `/etc/bacula/bacula restart` après un reboot du serveur.

Pour Bacula 1.38 installé depuis les sources :

1. Votre déclaration de `configure` doit comprendre :
   ```
   --with-dir-user=bacula
   --with-dir-group=bacula
   --with-sd-user=bacula
   --with-sd-group=disk
   --with-fd-user=root
   --with-fd-group=bacula
   ```
2. Ajoutez l'utilisateur "apache" au groupe "bacula" pour les systèmes Redhat ou Mandrake. Pour les SuSE, ajoutez l'utilisateur "wwwrun" au groupe "bacula"
3. Vérifier/corrigez le propriétaire de tous vos fichiers de Volume Bacula à root.bacula

Pour Bacula 1.36 ou 1.38 installé depuis des paquets RPM :

1. Ajoutez l'utilisateur "apache" au groupe "bacula" pour les systèmes Redhat ou Mandrake. Pour les SuSE, ajoutez l'utilisateur "wwwrun" au groupe "bacula"
2. Vérifier/corrigez le propriétaire de tous vos fichiers de Volume Bacula à root.bacula

bimagemgr installé depuis un paquet RPM > 1.38.9 se charge d'ajouter l 'utilisateur du serveur web au groupe bacula dans un script post-installation. Assurez-vous de modification la partie configurable de config.pm après l'installation du paquet RPM.

bimagemgr sera désormais capable de lire les fichiers de Volume mais ils ne sont toujours pas lisibles par tout le moned

Si vous utilisez bimagemgr depuis une autre machine (déconseillé), vous allez avoir besoin de changer les droits sur tous les fichiers de Volume en 644 pour mettre d'y accéder par exemple en NFS (entre autres). Cette approche ne doit être choisie que si vous êtes certain de la sécurité de votre environnement car dans ce cas les Volumes de sauvegarde peuvent être lus par n'importe qui.

1.16.2 Utilisation de bimagemgr

En appelant la programme à partir de votre navigateur, c'est-à-dire `http://localhost/cgi-bin/bimagemgr.pl`, vous obtiendrez un affichage comme à la figure **??** Le programme interroge la base de données d Bacula et affiche tous les Volumes avec la date de leur dernière écriture et la date à laquelle ils ont été gravés. Si un Volume doit être gravé (modifié depuis le dernier gravage), un bouton "Burn" sera affiché dans la dernière colonne à droite

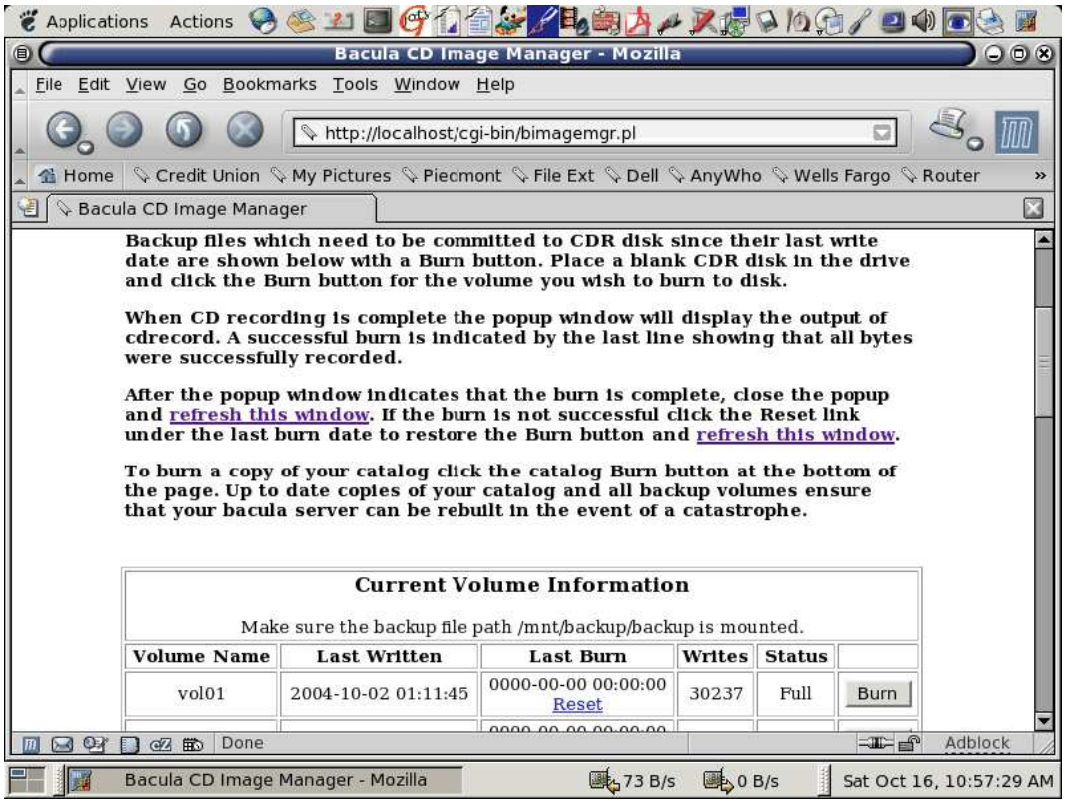

FIGURE 1.1 – Gestionnaires d'images CD Bacula

Insérez un support vierge dans votre graveur et cliquez sur le bouton "Burn". Ceci va ouvrir une fenêtre popup comme la figure **??** pour afficher la progression du gravage.

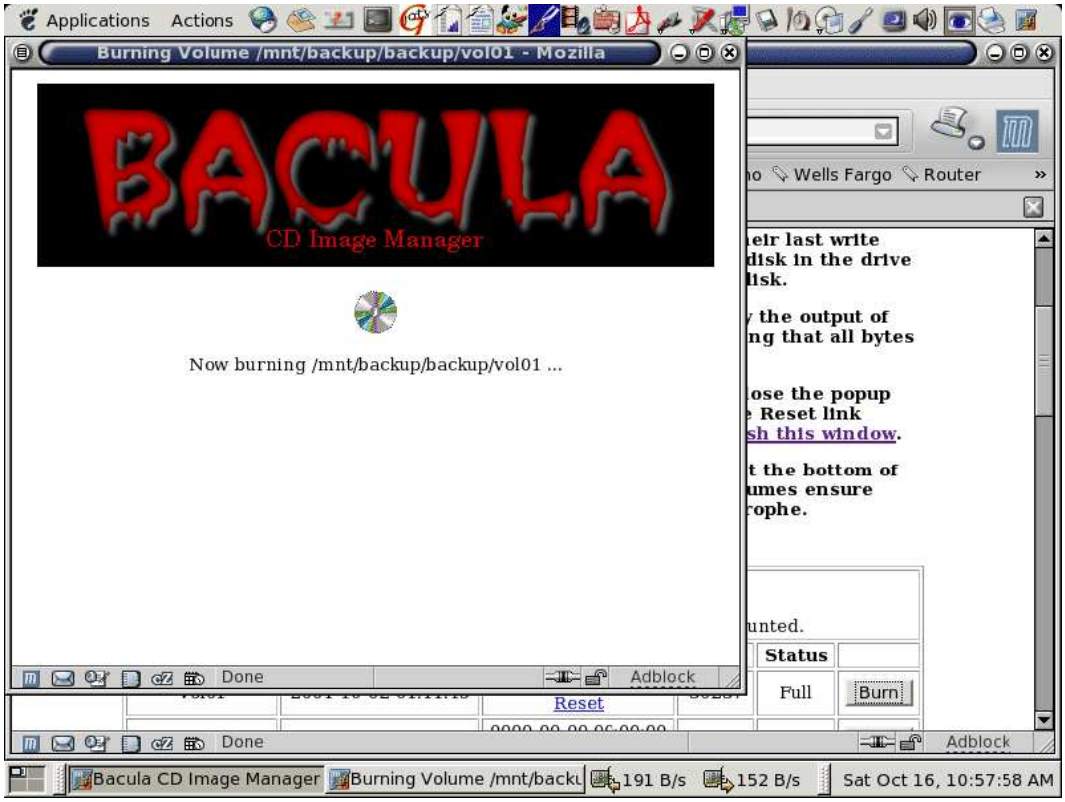

FIGURE 1.2 – Fenêtre de progression du gravage d'une image CD Bacula

A la fin du gravage, la fenêtre popup affichera les résultats de cdrecord comme vous pouvez le voir à la figure **??**. Fermez cette fenêtre et rafraîchissez la fenêtre principale. La date du dernier gravage aura été mise à jour et le bouton "Burn" disparaîtra pour ce volume. Si le gravage a échoué, vous pouvez réinitialiser la date de dernier gravage pour ce volume en cliquant sur le lien "Reset".

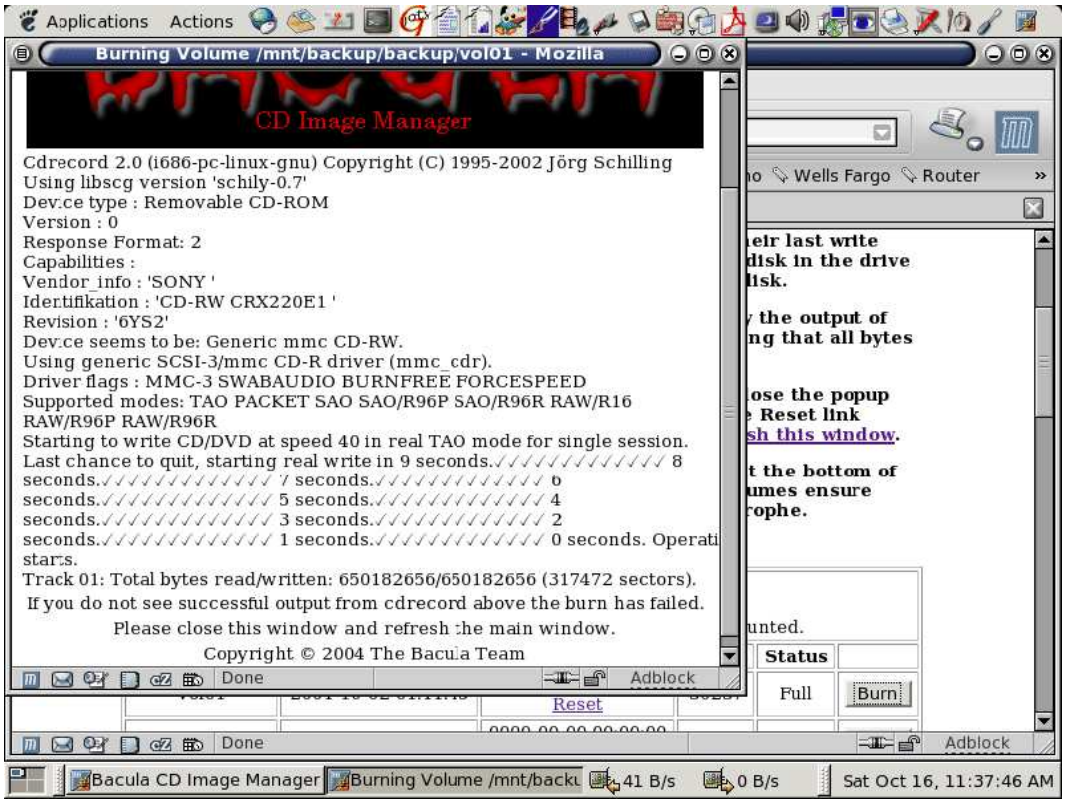

FIGURE 1.3 – Résultats du gravage d'une image CD Bacula

Sur la dernière ligne de la fenêtre principale, vous allez trouver deux boutons supplémentaires appelés "Burn Catalog" et "Blank CDRW". "Burn Catalog" fera une copie de votre catalogue Bacula sur disque. Si vous utilisez des supports réinscriptibles (CD-RW) au lieu de CD-R, le bouton "Blank CDRW" vous permettra d'effacer le disque avant de graver de nouveau dessus. La copie régulière de vos volumes et de votre catalogue vers des CD avec **bimagemgr** vous permet de tout reconstruire facilement en cas de désastre sur le serveur Bacula.

Chapitre 2

FAQ Création de paquets RPM Bacula

2.1 FAQ

2.1.1 Questions

1. **??**
2. **??**
3. **??**
4. **??**
5. **??**
6. **??**
7. **??**
8. **??**
9. **??**

2.1.2 Réponses

Comment compiler Bacula pour la plate-forme xxx ?

Le fichier de spec de Bacula contient des `define` permettant de le compiler pour plusieurs plates-formes :
– Red Hat 7.x (rh7), Red Hat 8.0 (rh8), Red Hat 9 (rh9),
– Fedora Core (fc1, fc3, fc4, fc5, fc6, fc7, fc8),
– Whitebox Enterprise Linux 3.0 (wb3),
– Red Hat Enterprise Linux (rhel3, rhel4, rhel5),
– Mandrake 10.x (mdk), Mandriva 2006.x (mdv),
– CentOS (centos3, centos4, centos5)
– Scientific Linux (sl3, sl4, sl5) and
– SuSE (su9, su10, su102, su103, su110).
La construction du paquet est contrôlée par un ensemble de `define` obligatoires au début du fichier spec. Ces `define` sont là pour déterminer les informations de dépendances contenues dans les paquets RPM résultants, ainsi que des options spécifiques à `configure`. Le `define` de plate-forme peut aussi être modifié directement dans le fichier de spec (par défault tous les `define` sont à 0 ou ”not set”). Par exemple, pour construire le paquet pour Redhat 7.x, il suffit de trouver la ligne suivante du fichier de spec :

```
%define rh7 0
```

et de la modifier de la façon suivante :

```
%define rh7 1
```

Une autre possibilité est de passer le define en ligne de commande de rpmbuild :

```
rpmbuild -ba --define "build_rh7 1" bacula.spec
rpmbuild --rebuild --define build_rh7 1" bacula-x.x.x-x.src.rpm
```

Comment décider quel sera le support des bases de données

Un autre `define` obligatoire décide du support des bases de données dans le binaire résultant, c'est un des suivants :
– `build_sqlite`
– `build_mysql`
– `build_postgresql`
Pour construire le paquet avec le support de MySQL, sans distinction de version, il faut modifier les lignes suivantes :

```
%define mysql 0
OR
%define mysql4 0
OR
%define mysql5 0
```

en :

```
%define mysql 1
OR
%define mysql4 1
OR
%define mysql5 1
```

dans le fichier de spec ou bien en les passant en ligne de commande à rpmbuild :

```
rpmbuild -ba --define "build_rh7 1" --define "build_mysql 1" bacula.spec
rpmbuild -ba --define "build_rh7 1" --define "build_mysql4 1" bacula.spec
rpmbuild -ba --define "build_rh7 1" --define "build_mysql5 1" bacula.spec
```

Quels autres defines peuvent être utilisés ?

Trois autres defines à noter sont `depkgs_version`, `docs_version` et `_rescuever`. Ces deux defines sont modifiés à chaque release et doivent correspondre à la version des sources utilisées pour construire les paquets. En temps normal, vous n'avez pas besoin de les omdifier. Voyez aussi la section "Build Options" plus bas pour les autres options de construction qui peuvent être passées en ligne de commande.

Je rencontre des erreurs de permissions quand j'essaie de construire les paquets. Dois-je être root ?

Non, vous n'avez pas besoin d'être root, et c'est en fait une bonne habitude de construire les paquets RPM en utilisateur non privilégié. Les paquets de Bacula sont prévus pour être construits en tant qu'utilisateur normal, mais vous devez effectuez quelques modifications à votre système pour que cela fonctionne. Si vous construisez les paquets sur votre propre système, la méthode la plus simple est d'ajouter des permissions en écriture à tout le monde sur les répertoires de construction (/usr/src/redhat/, /usr/src/RPM or /usr/src/packages). Pour ce faire, tapez les commandes suivantes en tant que root :

```
chmod -R 777 /usr/src/redhat
chmod -R 777 /usr/src/RPM
chmod -R 777 /usr/src/packages
```

Si vous travaillez sur un système partagé où vous ne pouvez pas utliser la méthode ci-dessus, vous devez créer l'arborescence ci-dessus dans votre répertoire personnel (home). Créez ensuite un fichier appelé `.rpmmacros` dans votre répertoire pesonnel (ou modifiez le fichier s'il existe déjà) pour avoir le contenu de fichier suivant :

```
%_topdir /home/myuser/redhat
%_tmppath /tmp
```

Une autre directive pratique pour empêcher la création de paquets RPM de debug peut être ajoutée à votre fichier `.rpmmacros` :

```
%debug_package %{nil}
```

Je construis mes propres RPMs mais sur toutes les plates-formes, j'ai une dépendance non résolue sur /usr/afsws/bin/pagsh.

C'est un shell de l'OpenAFS (Andrew File System). Si vous rencontrez cette erreur, c'est que vous avez choisi d'inclure le répertoire docs/examples dans votre paquet. Un des scripts d'exemple de ce répertoire est un script pagsh. Quand `rpmbuild` analyse les dépendances, il vérifie la présence de la ligne shebang de tous les scripts inclus dans le package, en plus des vérifications de librairies partagées. Pour ne pas avoir cette dépendance, il ne faut pas include le répertoire des exemples dans le paquet. Si vous rencontrez de problème, vous devez être en train de construire un paquet d'une très vieille version de Bacula, car les exemples ont été supprimés du paquet de documentation.

Je construis mes propres RPMs car vous ne les fournissez pas pour ma plate-forme. Puis-je publier mes paquets sur Sourceforge pour qu'ils servent à d'autres ?

Oui, les contributions de la part d'utilisateurs sont acceptées et bien sûr appréciées ! Consultez le répertoire `platforms/contrib-rpm` dans les sources pour plus de détails.

Existe-t'il une solution plus simple que d'utiliser toutes ces options en ligne de commmande ?

Oui, il y a un assistant graphique que vous pouvez utiliser pour construire le paquet src RPM. Vous trouverez dans les sources le script `platforms/contrib-rpm/rpm_wizard.sh`, il vous permettra de spécifier les options de construction en passant par une interface graphique GNOME. Ce script nécessite d'avoir zenity installé.

Je viens juste mettre à jour Bacula de la version 1.36.x en 1.38.x et le Director Daemon ne démarre plus. Il semble démarrer mais plante en silence, et j'ai une erreur "connection refused" quand je démarre la console.

A partir de la version 1.38, les paquets RPMS sont paramétrés pour exécuter les daemons Director et Storage en tant qu'utilisateur non privilégié (non-root). Le File Daemon est exécuté en tant qu'utilisateur root et groupe bacula, le Storage Daemon en tant qu'utilisateur bacula et groupe disk, et le Director en tant qu'utilisateur bacula et groupe bacula. Lors de la mise à jour, il faut modifier les droits sur certains fichier pour que tout fonctionne. Lancez les commandes suivantes en tant que root :

```
chown bacula.bacula /var/bacula/*
chown root.bacula /var/bacula/bacula-fd.9102.state
chown bacula.disk /var/bacula/bacula-sd.9103.state
```

Ensuite, si vous utilisez des volumes de type File plutôt que des bandes, ces fichiers devront appartenir à l'utilisateur bacula et au groupe bacula.

Il y a beaucoup de paquets RPM, duquel ai-je besoin pour quel rôle ?

Pour un serveur Bacula, vous devez choisir le paquet suivant le système de base de données que vous utiliserez pour le catalogue : c'est soit bacula-mysql, bacula-postgresql ou bacula-sqlite. Si votre système ne fournit pas de paquet pour mtx, vous devez installer bacula-mtx pour satisfaire la dépendance. Pour une machine client, vous devez seulement installer bacula-client. Pour soit un serveur ou uyn client, vous pouvez installer les consoles graphiques bacula-gconsole et/ou bacula-wxconsole. BAT (Bacula Administration Tool) est installé par le paquet bacula-bat. Pour finir, le paquet bacula-updatedb est requis seulement lors des mises à jour d'un serveur, de plus d'un niveau de révision de base de données.

2.2 Support for RHEL3/4/5, CentOS 3/4/5, Scientific Linux 3/4/5 and x86_64

Les exemples ci-dessous démontrent des constructions avec le support explicite de la RHEL4 et de la CentOS4. Le support de l'architecture x86_64 a également été ajouté.

Lancez la construction avec une de ces trois commandes :

```
rpmbuild --rebuild \
        --define "build_rhel4 1" \
        --define "build_sqlite 1" \
        bacula-1.38.3-1.src.rpm

rpmbuild --rebuild \
        --define "build_rhel4 1" \
        --define "build_postgresql 1" \
        bacula-1.38.3-1.src.rpm

rpmbuild --rebuild \
        --define "build_rhel4 1" \
        --define "build_mysql4 1" \
        bacula-1.38.3-1.src.rpm
```

Pour la CentOS, indiquez `-define "build_centos4 1"` à la place de rhel4. Pour la Scientific Linux, indiquez `-define "build_sl4 1"` à la place de rhel4.

Pour le support du 64 bits, ajoutez `-define "build_x86_64 1"`

2.3 Options de construction

Le fichier de spec supporte actuellement la construction sur les plateformes suivantes :

```
Red Hat builds
--define "build_rh7 1"
--define "build_rh8 1"
--define "build_rh9 1"

Fedora Core build
--define "build_fc1 1"
--define "build_fc3 1"
--define "build_fc4 1"
--define "build_fc5 1"
--define "build_fc6 1"
--define "build_fc7 1"
--define "build_fc8 1"
--define "build_fc9 1"

Whitebox Enterprise build
--define "build_wb3 1"
```

```
Red Hat Enterprise builds
--define "build_rhel3 1"
--define "build_rhel4 1"
--define "build_rhel5 1"

CentOS build
--define "build_centos3 1"
--define "build_centos4 1"
--define "build_centos5 1"

Scientific Linux build
--define "build_sl3 1"
--define "build_sl4 1"
--define "build_sl5 1"

SuSE build
--define "build_su9 1"
--define "build_su10 1"
--define "build_su102 1"
--define "build_su103 1"
--define "build_su110 1"
--define "build_su111 1"

Mandrake 10.x build
--define "build_mdk 1"

Mandriva build
--define "build_mdv 1"

MySQL support:
for mysql 3.23.x support define this
--define "build_mysql 1"
if using mysql 4.x define this,
currently: Mandrake 10.x, Mandriva 2006.0, SuSE 9.x & 10.0, FC4 & RHEL4
--define "build_mysql4 1"
if using mysql 5.x define this,
currently: SuSE 10.1 & FC5
--define "build_mysql5 1"

PostgreSQL support:
--define "build_postgresql 1"

Sqlite support:
--define "build_sqlite 1"

Build the client rpm only in place of one of the above database full builds:
--define "build_client_only 1"

X86-64 support:
--define "build_x86_64 1"

Supress build of bgnome-console:
--define "nobuild_gconsole 1"

Build the WXWindows console:
requires wxGTK >= 2.6
--define "build_wxconsole 1"

Build the Bacula Administration Tool:
requires QT >= 4.2
--define "build_bat 1"

Build python scripting support:
--define "build_python 1"

Modify the Packager tag for third party packages:
--define "contrib_packager Your Name <youremail@site.org>"

Install most files to /opt/bacula directory:
--define "single_dir_install 1"

Build the rescue files:
--define "build_rescue 1"
```

2.4 Problèmes d'installation de RPMs

Une fois qu'ils sont correctement construits, les paquets RPM s'installent en général sans problème. Toute fois, certains problèmes peuvent se déclarer au lancement des daemons :

– Mauvaises permissions sur /var/bacula : par défaut, les daemons Director Storage ne sont pas exécutés en root. Si /var/bacula appartient à root, il est possible que les daemons Director et Storage ne soient pas capables d'accéder à ce répertoire, alors que c'est leur répertoire de travail (Working Directory). Pour corriger celà, la méthode la plus simple est : `chown bacula:bacula /var/bacula`.
  Note : à partir de la version 1.38.8, le répertoire /var/bacula est créé avec des permissions en mode 770 et un propriétaire à root :bacula.
– Le Storage daemon ne peut pas accéder au lecteur de bandes : ceci peut arriver dans des anciennes versions de paquets RPM où le Storage Daemon fonctionnait sous l'identité userid bacula, groupe bacula. Il y a deux méthodes pour corriger ceci : la meilleure reste de modifier le script de démarrage /etc/init.d/bacula-sd pour que le Storage daemon soit lancé sous le groupe "disk". La seconde méthode est de changer les droits sur le device du lecteur de bande (habituellement /dev/nst0) pour que Bacula puisse y accéder. Vous devrez certainement aussi changer les permissions sur le device de contrôle SCSI, habituellement /dev/sg0. Les noms exacts de device dépendent de votre configuration, référez-vous au chapitre "Tape Testing" pour plus d'informations sur les devices.

Table des figures