Il surgit de la nuit et absorbe l'essence vitale de vos ordinateurs.

Kern Sibbald
Traduit de l'anglais par Ludovic Strappazon

3 janvier 2008
Ce mode d'emploi documente la version 1.38.11 (28 June 2006) de Bacula

Table des matières

Table des figures

Liste des tableaux

Qu'est-ce que Bacula?

Bacula est un jeu de programmes qui vous permet (ou à l'administrateur système) de faire des sauvegardes, restaurations, et vérifications des données d'un ordinateur sur un réseau hétérogène. Bacula peut fonctionner complètement sur un seul ordinateur. Il est capable de sauvegarder sur des supports variés, y compris disques et cartouches.

En termes techniques, il s'agit d'un programme de sauvegarde Client/Serveur. Bacula est relativement facile d'utilisation et efficace, tout en offant de nombreuses fonctions avancées de gestion de stockage qui facilitent la recherche et la restauration de fichiers perdus ou endommagés. Grâce à sa conception modulaire, Bacula est échelonnable depuis le simple système constitué d'un ordinateur, jusqu'au système de plusieurs centaines d'ordinateurs disséminés sur un vaste réseau.

Qui a besoin de Bacula?

Si vous utilisez actuellement un programme tel que **tar**, **dump**, ou **bru** pour sauvegarder vos données, et aimeriez une solution réseau, plus de flexibilité, ou les commodités d'un catalogue, Bacula vous procurera certainement les fonctions supplémentaires que vous recherchez. Cependant, si vous avez peu d'expérience des systèmes Unix ou si vous n'avez pas l'expérience d'un système de sauvegarde sophistiqué, nous ne vous recommandons pas l'utilisation de Bacula, car il est beaucoup plus difficile à installer et utiliser que **tar** ou **dump**.

Si vous attendez de Bacula qu'il se comporte comme les programmes simples mentionnés ci-dessus et qu'il écrive sur toute cartouche insérée dans le lecteur, vous éprouverez des difficultés à travailler avec Bacula. Bacula est conu pour protéger vos données en suivant les règles que vous spécifiez, ce qui signifie que la réutilisation d'une cartouche ne se fera qu'en dernier ressort. Il est possible de "contraindre" Bacula à écraser toute cartouche dans le lecteur, mais il est plus facile et plus efficace d'utiliser un outil plus basique pour ce genre d'opérations.

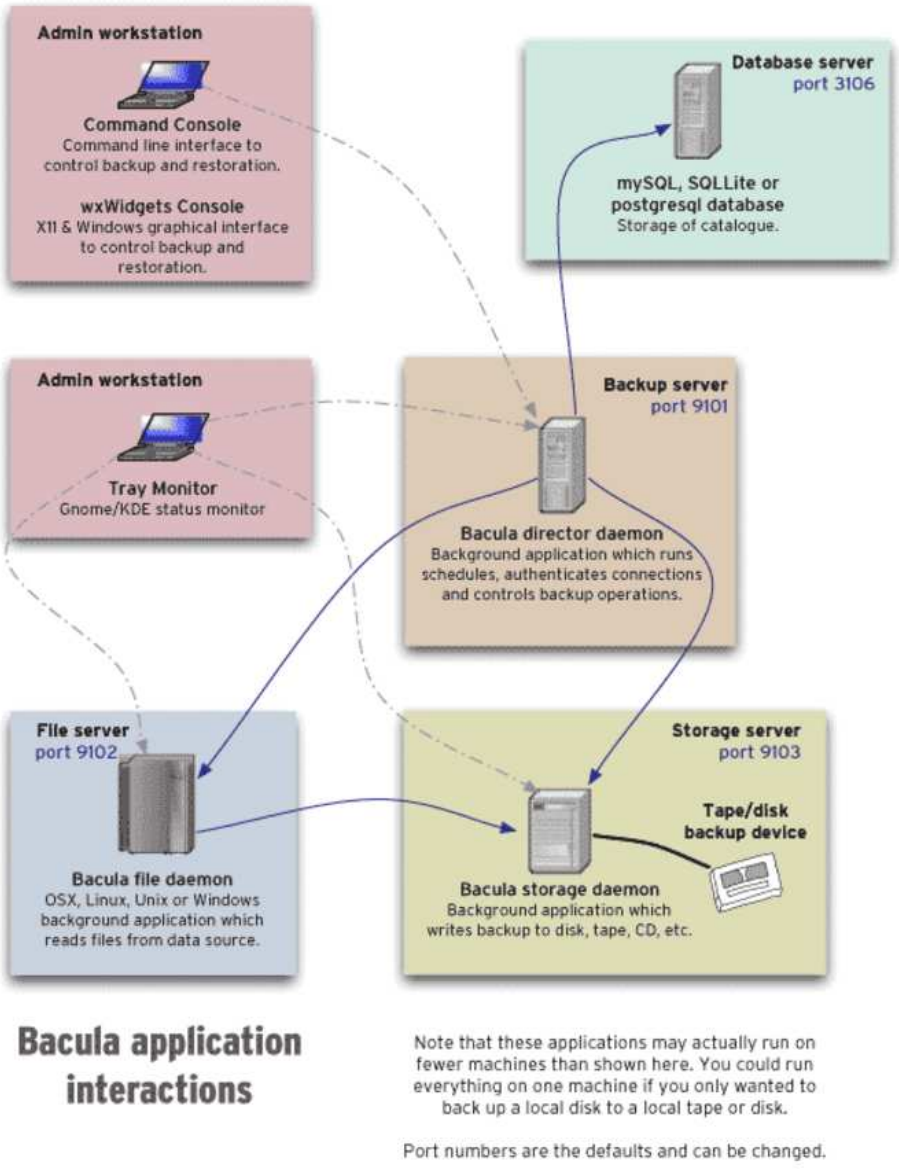
Si vous utilisez **Amanda** et aimeriez un programme de sauvegarde qui peut écrire sur plusieurs volumes (qui ne soit pas limité par la capacité de vos cartouches), Bacula peut certainement satisfaire vos besoins, d'autant que plusieurs de nos utilisateurs estiment que Bacula est plus simple à installer et utiliser que d'autres programmes équivalents.

Si vous utilisez actuellement un logiciel commercial sophistiqué tel que

Legato Networker, **ARCserveIT**, **Arkeia**, ou **PerfectBackup+**, vous pourriez être interessé par Bacula qui fournit la plupart de leurs fonctions et qui est un logiciel libre sous licence GNU version 2.

Composants ou Services de Bacula

Bacula est constitué des cinq composants ou services majeurs suivants :



(remerciements à Aristedes Maniatis pour ce schéma et le suivant)

– Le service **Bacula Director** est le programme qui supervise toutes les opérations de sauvegarde, restauration, vérification et archivage. L'administrateur système utilise le Bacula Director pour planifier les sauvegardes et restaurer les fichiers. Pour plus de détails, consultez la section concernant la conception du Daemon Director dans la documentation pour développeurs. Le Director est exécuté en tant que *daemon* ou service (c'est à dire en tche de fond).

- Le service **Bacula Console** est le programme qui permet à l'administrateur ou à l'utilisateur de communiquer avec le **Bacula Director** (voir ci-dessus). Actuellement, le service Bacula Console est disponible en trois versions. La première et la plus simple est d'exécuter le programme Console dans une fenêtre shell (i.e. interface TTY). La plupart des administrateurs système trouveront cette méthode parfaitement adéquate. La seconde version est une interface graphique GNOME qui est loin d'être complète, mais est tout à fait fonctionnelle puisqu'elle possède la plupart des possibilités de la Console shell. La troisième version est une interface graphique wxWidgets qui permet de sélectionner interactivement les fichiers à restaurer. Elle intègre la plupart des fonctionnalités de la console shell, permet la complétion automatique avec la touche tabulation, et fournit une aide instantanée relative à la commande que vous êtes en train de taper. Pour plus de détails, consultez la section Conception de la Console Bacula dans la documentation pour développeurs.
- Le service **Bacula File** (ou programme client) est le programme installé sur la machine à sauvegarder. Il est spécifique au système sur lequel il est exécuté et a la charge de fournir les attributs des fichiers et les données requis par le Director. Les Services File sont aussi chargés de la partie dépendant du système de fichiers lors de la restauration des attributs de fichiers et des données. Pour plus de détails, consultez le document sur la conception du File Daemon dans le Guide pour Developpeurs. Ce programme est exécuté en tant que service sur la machine à sauvegarder, et la documentation s'y réfère parfois en tant que Client (par exemple dans les fichiers de configuration de Bacula). En plus du File Daemon pour Unix/Linux, il existe un File Daemon pour Windows (usuellement distribué au format binaire). Le File Daemon Windows fonctionne sur toutes les versions actuelles de Windows (NT, 2000, XP, 2003 et peut-être aussi 98 et Me).
- Le service **Bacula Storage** est le programme qui transfère les données et les attributs de fichiers aux média physiques ou aux volumes et les restitue lors de restaurations. En d'autres termes, Le storage Daemon est responsable des opérations de lecture et d'écriture sur vos cartouches (ou autres média de stockage, comme par exemple des fichiers). Pour plus de détails consultez la Documentation Pour Développeurs sur la conception du Storage Daemon.
- Les services **Catalogue** ont pour tâche de maintenir à jour la base de données des index de fichiers et volumes pour tous les fichiers sauvegardés. Les services **Catalogue** permettent à l'administrateur système ou à l'utilisateur de localiser rapidement et restaurer n'importe quel fichier. Les services Catalogue de Bacula le placent dans une catégorie différente de programmes tels que tar et bru, puisque le catalogue Ba-

cula maintient un enregistrement de chaque volume utilisé, chaque job exécuté et chaque fichier sauvegardé ce qui permet des restaurations et une gestion de volumes efficaces. Bacula supporte actuellement trois bases de données différentes, MySQL, PostgreSQL, et SQLite. L'une des trois doit être choisie à la compilation de **Bacula**.

Les trois bases de données actuellement supportées (MySQL, PostgreSQL, ou SQLite) fournissent de nombreuses fonctions telles l'indexation rapide, requêtes arbitraires, et sécurité. Bien que nous prévoyions de supporter d'autres bases de données SQL majeures, l'implémentation actuelle s'interface seulement avec MySQL, PostgreSQL, et SQLite. Pour plus de détails consultez le document sur la conception des Services Catalogue .

Les RPMs pour MySQL et PostgreSQL font partie de la distribution Red Hat. Sinon, il est tout à fait aisé de les construire à partir des sources. Consultez le chapitre Installer et configurer MySQL ou Installer et configurer PostgreSQL de ce document pour plus de détails. Pour plus d'informations sur MySQL et PostgreSQL, consultez www.mysql.com ou www.postgresql.org.

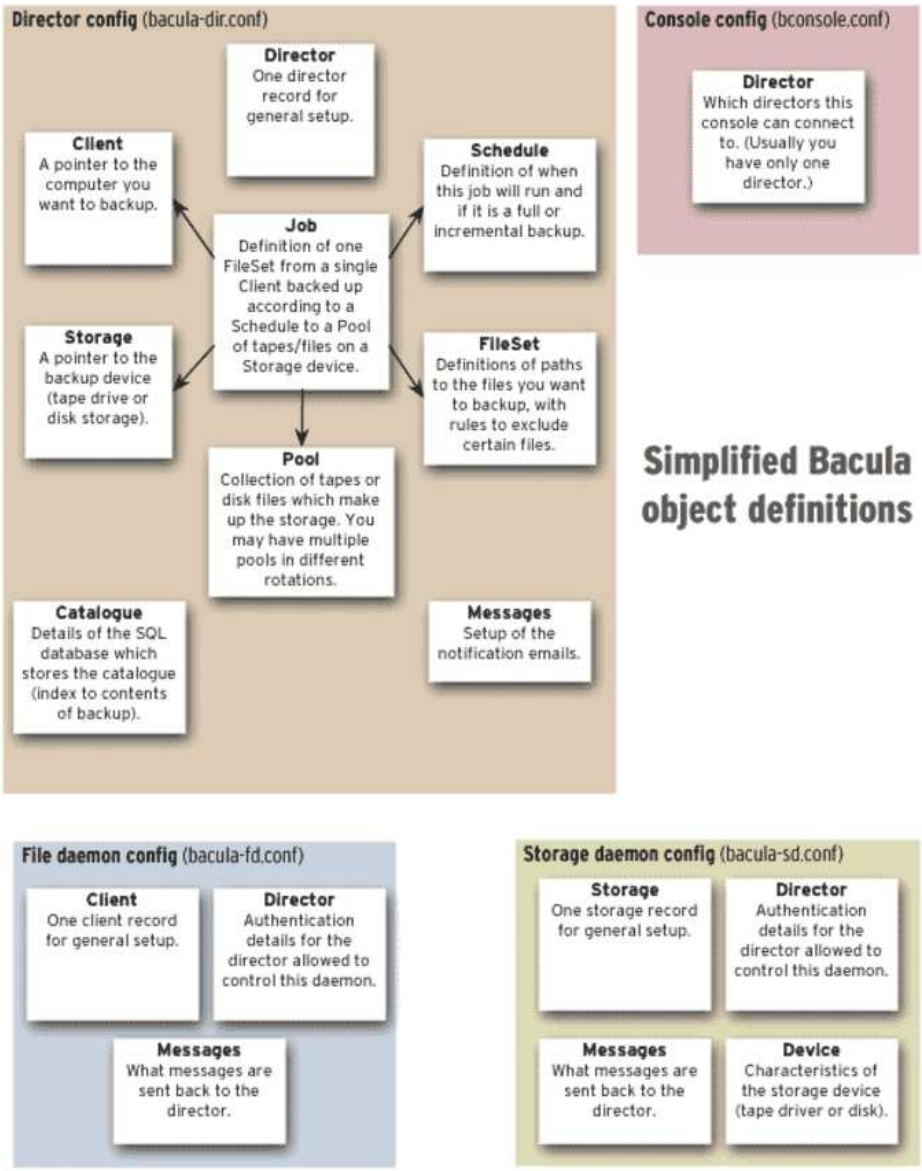
Configurer et construire SQLite est encore plus facile. Pour les détails de configuration de SQLite, consultez le chapitre Installer et Configurer SQLite de ce document.

– Le service **Bacula Monitor** est le programme qui permet à l'administrateur ou à l'utilisateur de contrôler le statut des *daemons* Bacula (**Bacula Directors**, **Bacula File Daemons** et **Bacula Storage Daemons**) (voir ci-dessus). Actuellement, la seule version disponible est une version GTK+, qui fonctionne avec Gnome et KDE (ainsi que tout gestionnaire de fenêtre qui respecte le standard system tray FreeDesktop.org).

Pour réaliser avec succès les opérations de sauvegarde et restauration, les quatre services suivants doivent être configurés et lancés : le Director Daemon, le File Daemon, le Storage Daemon et MySQL, PostgreSQL ou SQLite.

Configuration de Bacula

Pour que Bacula comprenne votre système, quels clients vous voulez sauvegarder et comment, vous devez créer un certain nombre de fichiers de configuration. La suite brosse un tableau de ces opérations.



Conventions utilisées dans ce document

Bacula est en constante évolution, par conséquent, ce manuel ne sera pas toujours en accord avec le code. Si un objet de ce manuel est précédé d'un astérisque (*), cela signifie que cette fonctionnalité particulière n'est pas implémentée. S'il est précédé d'un signe plus (+), cela indique que la fonction est peut-être partiellement implémentée.

Si vous lisez la version de ce manuel fournie avec les sources de Bacula, le paragraphe ci-dessus reste vrai. En revanche, si vous lisez la version en ligne : www.bacula.org/manual, veuillez garder à l'esprit que cette version décrit la version courante de développement de Bacula (celle du CVS) qui peut contenir des fonctionnalités qui n'existent pas dans la version "officielle". De même, il est généralement un peu à la traîne derrière le code.

Démarrage rapide

Pour faire fonctionner Bacula rapidement, nous vous recommandons de commencer par parcourir la section Terminologie ci-dessous, de passer rapidement en revue le chapitre suivant intitulé L'état actuel de Bacula, puis le Guide de démarrage rapide de Bacula, qui vous donnera une vue d'ensemble de la mise en oeuvre de Bacula . Après quoi vous devriez poursuivre avec le chapitre sur L'installation de Bacula, puis le chapitre Comment configurer Bacula, et finalement, le chapitre Exécuter Bacula.

Terminologie

Pour faciliter la communication autour de ce projet, nous fournissons ici les définitions de la terminologie que nous utilisons.

Administrateur La ou les personne(s) responsable(s) de l'administration du système Bacula.

Sauvegarde Nous utilisons ce terme pour un job Bacula qui sauvegarde des fichiers.

Fichier Bootstrap (Bootstrap File) Le bootstrap est un fichier ASCII qui contient, sous une forme compacte, les commandes qui permettent à Bacula ou à l'utilitaire autonome **bextract** de restaurer les contenus d'un ou plusieurs volumes, par exemple, l'état courant d'un système qui vient d'être sauvegardé. Avec un fichier bootstrap, Bacula peut restaurer votre système sans catalogue. Vous pouvez créer un fichier bootstrap depuis un catalogue pour extraire le fichier que vous voulez.

Catalogue Le catalogue est utilisé pour stocker des informations sommaires concernant les Jobs et Clients, les fichiers qui ont été sauvegardés ainsi que le ou les volume(s) où ils ont été sauvegardés. L'information stockée dans le catalogue permet à l'administrateur ou aux utilisateurs de déterminer quels jobs ont été exécutés, leur statut, ainsi que d'importantes caractéristiques de chaque fichier sauvegardé. Le catalogue est une ressource en ligne, mais ne contient pas les données pour les

fichiers sauvegardés. La plupart des informations stockées dans le catalogue le sont aussi sur les volumes de sauvegarde (i.e. cartouches). Bien sur, les cartouches auront aussi une copie du fichier en plus de ses attributs (voir ci-dessus).

La fonction Catalogue est de celles qui distinguent Bacula de simples programmes de sauvegarde et archivage tels que **dump** et **tar**.

Client Dans la terminologie de Bacula, le mot Client désigne une machine sauvegardée, et est synonyme de File service ou File Daemon. Nous nous y référons assez souvent par "le FD". Un client est défini dans une ressource de fichier de configuration.

Console Le programme qui interface le Director, permettant à l'administrateur de contrôler Bacula.

Daemon Terminologie Unix pour un programme toujours présent en arrière plan pour prendre en charge une tâche donnée. Sur les systèmes Windows, ainsi que certains Linux, les *daemons* sont appelés **Services**.

Directive Le terme directive est utilisé pour désigner une entrée ou enregistrement à l'intérieur d'une ressource dans un fichier de configuration qui définit un élément spécifique. Par exemple, la directive **Name** définit le nom de la ressource.

Director Le principal *daemon* serveur de Bacula qui planifie et dirige toutes les opérations de Bacula. Occasionnellement, nous le désignons par "le DIR".

Differentielle (Differential) Une sauvegarde qui inclut tous les fichiers modifiés depuis le lancement de la dernière sauvegarde complète (Full). Notez que d'autres logiciels de sauvegarde peuvent définir ceci différemment.

Attributs de fichiers Les Attributs de fichiers sont toutes les informations nécessaires au sujet d'un fichier pour l'identifier, et toutes ses propriétés telles taille, date de création, date de modification, permission, etc. En principe, les attributs sont intégralement manipulés par Bacula de sorte que l'utilisateur n'a jamais à s'en préoccuper. Les attributs n'incluent pas les données du fichier.

File Daemon Le *daemon* exécuté sur l'ordinateur client à sauvegarder. Il est aussi désigné par Service Fichier (File Service) et parfois Service Client ou FD.

FileSet Un FileSet est une ressource d'un fichier de configuration qui définit les fichiers à sauvegarder. Il consiste en une liste de fichiers ou répertoires inclus, une liste de fichiers ou répertoires exclus et la façon dont les fichiers seront stockés (compression, chiffrement, signatures). Pour plus de détails consultez le paragraphe Définition de la Ressource FileSet dans le chapitre Director de ce document.

Incrementale Une sauvegarde qui inclut tous les fichiers modifiés depuis le lancement de la dernière sauvegarde complète (Full), différentielle, ou incrémentale. Normalement spécifié dans la directive **Level** (niveau) dans la définition de la ressource Job, ou dans une ressource Schedule.

Job Un Job Bacula est une ressource de configuration qui définit le travail que Bacula doit effectuer pour sauvegarder ou restaurer un client particulier. Un Job consiste en un **Type**, (Type : backup, restore, verify, etc.), un **Niveau** (Level : full, incremental, ...), un **FileSet**, et un lieu de **Stockage** où écrire les fichiers (Storage device, Media Pool). Pour plus de détails consultez le chapitre Définition des Ressources Job de ce document.

Monitor Le programme qui s'interface avec chacun des *daemons* pour permettre à l'utilisateur ou à l'administrateur de surveiller le statut de Bacula.

Resource Une ressource est une partie d'un fichier de configuration qui définit une unité spécifique d'information disponible pour Bacula. Par exemple, la ressource **Job** définit toutes les propriétés d'un Job spécifique : nom, schedule (planification), volume pool, type de sauvegarde, niveau de sauvegarde, etc.

Restore Une Restore est une ressource de configuration qui décrit l'opération de restauration d'un fichier (perdu ou endommagé) depuis un medium de sauvegarde. C'est l'opération réciproque d'une sauvegarde, sauf que, dans la plupart des cas, une restauration concernera un petit ensemble de fichiers tandis qu'une sauvegarde concerne le plus souvent l'ensemble des fichiers d'un système. Bien sur, après une défaillance de disque(s), Bacula peut être appelé à effectuer une restauration complète de tous les fichiers qui étaient sur le système.

Schedule Un Schedule est une ressource de configuration qui définit le moment de l'exécution du Job Bacula. Pour utiliser un schedule, la ressource Job se réfère au nom du Schedule. Pour plus de détails, consultez la Définition de la ressource Schedule dans le chapitre Director de ce document.

Service Terminologie Windows pour désigner un *daemon* – Voir plus haut. Elle est maintenant fréquemment utilisée dans les environnements Unix aussi.

Adresses de stockage Les informations retournées par les Storage services qui localisent de façon unique les fichiers sur un medium de sauvegarde. Elles consistent en deux parties : l'une appartient à chaque fichier sauvegardé, l'autre à l'ensemble du Job. Normalement, cette information est sauvegardée dans le catalogue de sorte que l'utilisateur n'a pas besoin de connaissances particulières des adresses de stockage. L'adresse de stockage inclut les attributs de fichiers (voir plus haut) ainsi que la localisation unique de l'information sur le volume de sau-

vegarde.

Storage Daemon Le Storage Daemon, parfois désigné par "SD" est le programme qui écrit les attributs et les données sur un Volume de Stockage (Storage Volume) (Usuellement une cartouche ou un disque).

Session Désigne en principe le dialogue interne entre le File Daemon et le Storage Daemon. Le File Daemon ouvre une **session** avec le Storage Daemon pour sauvegarder un Fileset, ou pour le restaurer. Une session est associée à un et un seul Job Bacula (voir plus haut).

Verify Il s'agit d'un job qui compare les attributs du fichier actuel aux attributs qui ont été préalablement stockés dans le catalogue Bacula. Cette fonction peut être utilisée pour détecter les modifications de systèmes de fichiers critiques, à la façon de **Tripwire**. L'un des avantages majeurs de l'utilisation de Bacula pour cette tâche est que sur la machine que vous voulez protéger, vous pouvez n'exécuter que le File Daemon. Le Director, le Storage Daemon et le catalogue peuvent résider sur une autre machine. Par conséquent si votre serveur est un jour compromis, il est peu probable que la base de données de vérification ait été trifouillée.

Verify peut aussi être utilisé pour s'assurer que les données les plus récemment écrites sur un volume sont cohérentes avec ce qui figure dans le catalogue (c-à-d il compare les attributs de fichiers), ou encore, confronter le contenu du volume aux fichiers originaux sur le disque.

***Archive** Une opération d'archivage est effectuée après une sauvegarde, et consiste en l'exclusion des volumes sur lesquels les données sont sauvegardées de l'utilisation courante. Ces volumes sont marqués "Archived", et ne sont plus utilisés pour sauvegarder des fichiers. Tous les fichiers contenus sur un Volume Archive sont supprimés du catalogue. PAS ENCORE IMPLEMENTE.

***Update** Une opération Update synchronise les fichiers du système distant sur ceux du local. Ceci est l'équivalent d'une fonctionnalité **rdist**. PAS ENCORE IMPLEMENTE.

Période de rétention Bacula reconnait plusieurs sortes de périodes de rétention. Les plus importantes sont la période de rétention des fichiers, la période de rétention des jobs et la période de rétention des volumes. Chacune de ces périodes de rétention désigne la durée pendant laquelle l'enregistrement spécifique sera conservé dans le catalogue. Ceci ne doit pas être confondu avec le temps pendant lequel les données sauvegardées sur un volume sont valides.

La période de rétention des fichiers détermine la durée pendant laquelle les enregistrements concernant les fichiers seront gardés dans le catalogue. Cette période est importante car le volume des enregistrements relatifs aux fichiers occupe, de loin, le plus d'espace dans la base de données. Par conséquent, vous devez vous assurer qu'un "élagage"

(NDT : pruning) régulier de ces enregistrements est effectué. (Voir la commande **retention** de la Console pour plus de détails sur ce sujet).
La période de rétention des jobs est la durée pendant laquelle les enregistrements relatifs aux jobs seront conservés dans le catalogue. Notez que tous les enregistrements relatifs aux fichiers sont attachés aux jobs qui ont sauvegardé ces fichiers. Les enregistrements relatifs aux fichiers peuvent être purgés tout en conservant ceux relatifs aux jobs. Dans ce cas, l'information concernant les jobs exécutés restera disponible, mais pas les détails des fichiers sauvegardés. Normalement, lorsqu'un job est purgé, tous les enregistrements concernant les fichiers qu'il a sauvegardé le sont aussi.
La période de rétention des volumes est la durée minimale de conservation d'un volume avant qu'il ne soit réutilisé. Bacula n'effacera, en principe, jamais un volume qui contient la seule copie de sauvegarde d'un fichier. Dans les conditions idéales, le catalogue maintiendrait les entrées pour tous les fichiers sauvegardés pour tous les volumes courants. Une fois qu'un volume est écrasé, les fichiers qui étaient sauvegardés dessus sont automatiquement effacés du catalogue. Cependant, s'il y a un très gros pool de volumes ou si un volume n'est jamais écrasé, le catalogue pourrait devenir énorme. Pour maintenir le catalogue dans des proportions gérables, les informations de sauvegarde devraient être supprimées après une période de rétention des fichiers définie.

Scan Une opération de scan consiste en un balayage du contenu d'un volume ou d'une série de volumes. Ces volumes et les informations concernant les fichiers qu'ils contiennent sont restaurés vers le catalogue Bacula. Une fois ces informations restaurées, les fichiers sauvegardés sur ces volumes pourront être aisément restaurés. Cette fonction est particulièrement utile si certains volumes ou jobs ont dépassé leur période de rétention et ont été élagués ou purgés du catalogue. Le balayage des données des volumes est effectué en utilisant le programme **bscan**. Consultez la section bscan du chapitre sur les utilitaires Bacula de ce manuel pour plus de détails.

Volume Un volume est une unité d'archivage, usuellement une cartouche ou un fichier nommé sur disque où Bacula stocke les données pour un ou plusieurs jobs de sauvegarde. Tous les volumes Bacula ont un label unique (logiciel) écrit sur le volume par Bacula afin qu'il puisse être assuré de lire le bon volume. (En principe, il ne devrait pas y avoir de confusion avec des fichiers disques, mais avec des cartouches, il est facile de monter la mauvaise).

Ce que Bacula n'est pas

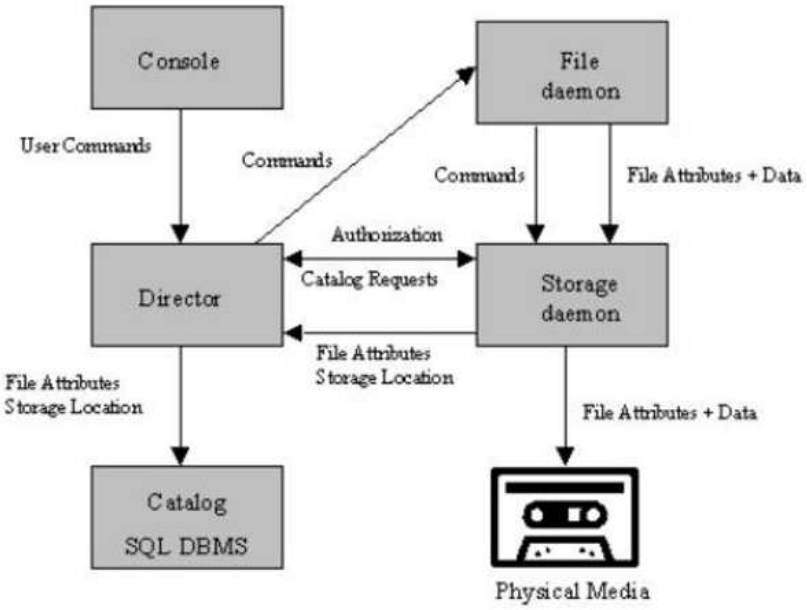
Bacula est un programme de sauvegarde, restauration et vérification, ce n'est pas un système complet de disaster recovery à lui seul, mais il peut en être une partie clef si vous planifiez soigneusement et suivez les instructions incluses dans le chapitre Plan de reprise d'activité avec Bacula de ce manuel.

Avec la planification appropriée, telle que décrite dans le chapitre sur le plan de reprise d'activité, **Bacula** peut devenir la pièce centrale de votre plan de reprise d'activité. Par exemple, si vous avez créé un(e) disque(tte) boot d'urgence et un(e) disque(tte) de secours Bacula pour sauvegarder les informations de partitionnement courantes de votre disque dur, et maintenu un jeu de sauvegardes complet de votre système, il est possible de reconstruire complètement votre système "depuis le métal brut" (NDT : From bare metal) .

Si vous avez utilisé la directive **WriteBootstrap** dans votre job ou quelque autre moyen pour sauvegarder un fichier bootstrap valide, vous pourrez l'utiliser pour extraire les fichiers nécessaires (sans utiliser le catalogue et sans chercher manuellement les fichiers à restaurer).

Interactions entre les services Bacula

Le diagramme fonctionnel suivant montre les interactions typiques entre les services Bacula pour un job de type sauvegarde. Chaque bloc représente en général un processus séparé (normalement un *daemon*). En général, le director surveille le flux d'informations. Il maintient aussi le catalogue.


Console
File daemon
User Commands
Commands
Commands
File Attributes + Data
Authorization
Catalog Requests
Director
Storage daemon
File Attributes
Storage Location
File Attributes
Storage Location
File Attributes + Data
Catalog
SQL DBMS
Physical Media

L'état actuel de Bacula

En d'autres termes, ce qui est et ce qui n'est pas actuellement implémenté et fonctionnel.

Ce qui est implémenté

- Job Control
  - Sauvegarde/restauration par le réseau avec un Director centralisé.
  - Scheduler interne pour le lancement automatique des Jobs.
  - Programmation de plusieurs Jobs à la même heure.
  - Execution simultanée d'un Job ou plusieurs Jobs.
  - Séquencement des Jobs selon une hiérarchie de priorités.
  - Console d'interfaage avec le Director permettant un contrle total. La console est disponible en version shell ou en mode graphique GNOME et wxWidget. Notez que pour l'instant, la version GNOME n'offre que trs peu de fonctionnalits supplmentaires par rapport la console shell.
- Scurit
  - Verification des fichiers prcdemment rfrencs offreant des possibilits la Tripwire (Vrification de l'intgrit du systme).
  - Authentification par change de mots de passe CRAM-MD5 entre chaque composant (*daemon*).
  - Chiffrement TLS (ssl) entre chaque composant.
  - Calcul de signatures MD5 ou SHA1 des fichiers sauvegards sur demande.
- Fonctionnalits lies aux restaurations
  - Restauration d'un ou plusieurs fichiers sélectionnés interactivement parmi les fichiers de la dernière sauvegarde ou ceux d'une sauvegarde antérieure à une date et heure données.
  - Restauration d'un système complet "depuis le métal brut". Cette opération est largement automatisée pour les systèmes Linux et partiellement pour les Solaris. Consultez le Plan de Reprise d'activité avec Bacula. Selon certains utilisateurs, la restauration "depuis le métal brut" fonctionne aussi pour les systèmes Win2K/XP.
  - Listage et restauration des fichiers avec les outils autonomes **bextract**. Entre autres choses, ceci permet l'extraction de fichiers quand Bacula et/ou le catalogue ne sont pas disponibles. Notez :

La méthode recommandée pour restaurer des fichiers est d'utiliser la commande restore dans la Console. Ces programmes sont conçus pour une utilisation en dernier recours.
    - Possibilité de régénérer le catalogue par balayage des volumes de sauvegarde grâce au programme **bscan**.
- Catalogue SQL
    - Fonctions de base de données (catalogue) pour les informations concernant les volumes, pools, jobs et fichiers sauvegardés.
    - Support pour des catalogues de type SQLite, PostgreSQL, et MySQL.
    - Requêtes utilisateur arbitraires sur les bases de données SQLite, PostgreSQL et MySQL.
- Gestion avance des pools et volume
    - Marquage (label) des Volumes pour prévenir tout écrasement accidentel (au moins par Bacula).
    - Un nombre quelconque de Jobs et Clients peuvent être sauvegardés sur un Volume unique. Cela signifie que vous pouvez sauvegarder et restaurer des machines Linux, Unix, Sun, et Windows sur le même volume.
    - Sauvegardes multi-volumes. Lorsqu'un Volume est plein, **Bacula** réclame automatiquement le volume suivant et poursuit la sauvegarde.
    - Gestion de librairie par Pools et Volumes offrant beaucoup de flexibilité dans la gestion des volumes (par exemple, groupes de volumes mensuels, hebdomadaires, quotidiens ou différenciés par client,...).
    - Format d'écriture de données sur les volumes indépendant des machines. Les clients Linux, Solaris, et Windows peuvent tous être sauvegardés sur le même volume si désiré.
    - Prise en charge flexible des messages incluant le routage des messages depuis n'importe quel *daemon* vers le Director pour un reporting automatique par e-mail.
    - Possibilité de mettre les données sur un tampon disque (data spooling) lors des sauvegardes avec criture sur cartouche asynchrone. Ceci prvient les arrts et redmarrage (NDT : "shoe shine") des lecteurs, surtout lors des incrmentales et diffrentielles.
- Support avanc pour la plupart des priphriques de stockage
    - Support pour les librairies de sauvegarde via une simple interface shell capable de s'interfacer avec pratiquement n'importe quel programme autochargeur.
    - Support pour les librairies équipées de lecteurs de codes barres – marquage (labeling) automatique selon les codes barres.

    - Support pour les librairies à magasins multiples, soit par l'utilisation des codes barres, soit par lecture des cartouches.
    - Support pour les librairies avec plusieurs lecteurs.
    - Sauvegardes/restaurations "Raw device". Les restaurations doivent alors s'effectuer vers le même support physique que la sauvegarde.
    - Tous les blocs de données des volumes (approx 64K bytes) contiennent une somme de contrôle.
- Support pour de nonbreux systmes d'exploitation
    - Programmé pour prendre en charge des noms de fichiers et messages arbitrairement longs.
    - Compression GZIP fichier par fichier effectuée, si activée, par le programme Client avant le transfert sur le réseau.
    - Sauvegarde et restaure les POSIX ACLs.
    - Liste d'accès à la console qui permet de restreindre l'accès des utilisateurs à leurs données seulement.
    - Support pour sauvegarde et restauration de fichiers de plus de 2GB.
    - Support pour les machines 64 bit, e.g. amd64.
    - Possibilité de chiffrer les communications entre les *daemons* en utilisant stunnel.
    - Support des tiquettes (labels) de cartouches ANSI et IBM.
    - Support des noms de fichiers Unicode (exemple : chinois) sur les machines Win32 depuis la version 1.37.28.
    - Sauvegarde cohrente des fichiers ouverts sur les systmes Win32 (WinXP, Win2003 mais pas Win2000), par l'utilisation de Volume Shadow Copy (VSS).
- Divers
    - Implémentation multi-thread.
    - Un fichier de configuration comprhensible et extensible pour chaque *daemon*.

Avantages de Bacula sur d'autres programmes de sauvegarde

- Du fait qu'il y a un client pour chaque machine, vous pouvez sauvegarder et restaurer des clients de tous types avec l'assurance que tous les attributs de fichiers sont convenablement sauvegardés et restaurés.
- Il est aussi possible de sauvegarder des clients sans aucun logiciel client en utilisant NFS ou Samba. Cependant, nous recommandons d'exécuter, si possible, un File Daemon client sur chaque machine à sauvegarder.

- Bacula prend en charge les sauvegardes multi-volumes.
- Une base de données complète aux standards SQL de tous les fichiers sauvegardés. Ceci permet une vue en ligne des fichiers sauvegardés sur n'importe quel volume.
- Elagage automatique du catalogue (destruction des anciens enregistrements), ce qui simplifie l'administration de la base de données.
- N'importe quel moteur de base de données SQL peut être utilisé, ce qui rend Bacula très flexible.
- La conception modulaire, mais intégrée rend Bacula très échelonnable.
- Puisque Bacula utilise des *daemons* fichier clients, toute base de données, toute application peut être arrétée proprement, puis redémarrée par Bacula avec les outils natifs du système sauvegardé (le tout dans un Job Bacula).
- Bacula intègre un Job Scheduler.
- Le format des volumes est documenté et il existe de simples programmes C pour le lire/écrire.
- Bacula utilise des ports TCP/IP bien définis (enregistrés) – pas de rpcs, pas de mémoire partagée.
- L'installation et la configuration de Bacula est relativement simple comparée à d'autres produits comparables.
- Selon un utilisateur, Bacula est aussi rapide que la grande application commerciale majeure.
- Selon un autre utilisateur, Bacula est quatre fois plus rapide qu'une autre application commerciale, probablement parce que cette application stocke ses informations de catalogue dans un grand nombre de fichiers plutôt que dans une base SQL comme le fait Bacula.
- Au lieu d'une interface d'administration graphique, Bacula possde une interface shell qui permet l'administrateur d'utiliser des outils tels que ssh pour administrer n'importe quelle partie de Bacula depuis n'importe o.
- Bacula dispose d'un CD de secours pour les systmes Linux dots des fonctionnalits suivantes :
  - Vous le gnrez sur votre propre systme d'un simple make suivi de make burn.
  - Il utilise votre noyau.
  - Il capture vos paramtres de disques et gnre les scripts qui vous permettront de repartitionner automatiquement vos disques et de les formater pour y remettre ce qui s'y trouvait avant le dsastre.
  - Il comporte un script qui redmarrera votre rseau (avec l'adresse IP correcte).
  - Il comporte un script qui monte automatiquement vos disques durs.

– Il comporte un Bacula FD complet statiquement li.
– Vous pouvez aisment y ajouter des donnes ou programmes additionnels.

Restrictions de l'implémentation actuelle

– Les chemins et noms de fichiers de longueur suprieure 260 caractres sur les systmes Win32 ne sont pas supports. Il sont sauvegards mais ne peuvent tre restaurs. L'utilisation de la directive **Portable=yes** dans votre FileSet permet de restaurer ces fichiers vers les systmes Unix et Linux. Les noms de fichiers longs seront implments dans une version ultrieure.
– Si vous avez plus de 4 billions de fichiers enregistrés dans votre catalogue, la base de données FileId atteindra probablement ses limites. Ceci est une base de données monstrueuse mais possible. A un certain stade, les champs FileId de Bacula passeront de 32 bits à 64 et ce problème disparaîtra. En attendant, un palliatif satisfaisant consiste à utiliser plusieurs bases de données
– Les fichiers supprimés après une sauvegarde full sont inclus dans les restaurations.
– Les sauvegardes diffrentielles et incrmentales de Bacula se basent sur les time stamps. Par consquent, si vous dplacez des fichiers d'un rpertoire existant ou un rpertoire complet appartenant un FileSet aprs une Full, ces fichiers ne seront probablement pas sauvegards par une incrmentale, car ils seront encore marqus des anciennes dates. Vous devez explicitement mettre jour ces dates sur tous les fichiers dplacs. La correction de ce dfaut est en projet.
– Les Modules Systme de Fichiers (routines configurables pour sauvegarder/restaurer les fichiers spciaux) ne sont pas encore implments.
– Le chiffrement des donnes sur les volumes n'est pas implment.
– Bacula ne peut restaurer automatiquement les fichiers d'un job depuis deux ou plusieurs priphriques de stockage diffrents. Si vous un mme job utilise plusieurs priphriques ou plusieurs types de mdia distincts, le processus de restauration ncessitera certaines interventions manuelles.

Limitations ou Restrictions inhérentes à la conception

– Les noms (tels que resource names, Volume names, ...) définis dans les fichiers de configuration de Bacula sont limités à un nombre fixé de caractères. Actuellement, la limite est définie à 127 caractères. Notez que ceci ne concerne pas les noms de fichiers qui peuvent être arbitrairement longs.

– Sur les machines Win32, les noms de fichiers sont limits 260 caractres par l'API non-Unicode Windows que nous utilisons. Il est prvu de lever cette limitation en basculant sur l'API Unicode.

Caractéristiques système générales indispensables à Bacula

Caractéristiques système générales indispensables à Bacula

- **Bacula** a été compilé et exécuté sur les systèmes Linux RedHat, Mandriva, SUSE, Debian et Gentoo, sur FreeBSD, et Solaris.
- Il requiert GNU C++ version 2.95 ou supérieur pour compiler. Vous pouvez essayer avec d'autres compilateurs et des versions plus anciennes, mais vous serez seuls. Nous avons compilé et utilisé avec succès Bacula sur RH8.0/RH9/RHEL 3.0 avec GCC 3.2. Note, en général GNU C++ est un paquet séparé (e.g. RPM) de GNU C, et vous devrez avoir les deux. Sur les systèmes RedHat, le compilateur C++ fait partie du paquet RPM **gcc-c++**.
- Certains paquets tiers sont nécessaires à **Bacula**. Excepté pour MySQL et PostgreSQL, ils peuvent tous être trouvés dans les distributions **depkgs** et **depkgs1**.
- Si vous voulez construire les binaires Win32, vous aurez besoin du compilateur Microsoft Visual C++ (ou Visual Studio). Bien que tous les composants compilent (la console produit quelques messages d'alertes), seul le File Daemon a été testé.
- **Bacula** requiert une bonne implémentation fonctionnelle des pthreads. Ce n'est pas le cas sur certains systèmes BSD.
- Le code source a été écrit dans un esprit de portabilité et est le plus souvent compatible POSIX. Ainsi le portage sur chaque système d'exploitation compatible POSIX est relativement aisé.
- Le programme GNOME Console est developpé et testé sous GNOME 2.X. Il s'exécute aussi sous GNOME 1.4 mais cette version est dépréciée et n'est plus maintenue.
- Le programme wxWidgets Console est developpé et testé avec la dernière version stable de wxWidgets (2.4.2). Il fonctionne bien avec la version Windows et GTK+-1.x de wxWidgets, ainsi que sur les autres plateformes supportées par wxWidgets.
- Le programme Tray Monitor est developpé pour GTK+-2.x. Il nécessite Gnome >=2.2, KDE >=3.1 ou un gestionnaire de fenêtre supportant le standard systemtray de FreeDesktop.
- Si vous voulez permettre l'édition en ligne de commande et l'historique, il vous faudra /usr/include/termcap.h et l'une des bibliothèques termcap ou ncurses chargée (libtermcap-devel ou ncurses-devel).
- Si vous voulez utiliser des DVD en guise de media de sauvegarde, vous devrez télécharger les dvd+rw-tools 5.21.4.10.8, appliquer

le patch pour rendre ces outils compatibles avec Bacula, puis les compiler et installer. N'utilisez pas les dvd+rw-tools fournis par votre distribution, ils ne fonctionneront pas avec Bacula.

Systèmes d'exploitation supportés

- Systèmes Linux (compilé et testé sur RedHat Enterprise Linux 3.0).
- Si vous avez un système Red Hat récent exécutant le noyau 2.4.x et si vous avez le répertoire **/lib/tls** installé sur votre système (par défaut normalement), **Bacula ne fonctionnera pas correctement** Ceci est dû à la nouvelle bibliothèque pthreads qui est défectueuse. Vous devez supprimer ce répertoire avant d'exécuter Bacula, ou vous pouvez simplement le renommer en **/lib/tls-broken** puis redémarrer votre machine (une des rares occasions où; Linux doit être redémarré). Si vous ne souhaitez pas déplacer/renommer /lib/tls, une autre alternative est de placer la variable d'environnement "LD_ASSUME_KERNEL=2.4.19" avant d'exécuter Bacula. Pour cette option, vous n'avez pas besoin de redémarrer, et tous les programmes autres que **Bacula** continueront d'utiliser **/lib/tls**.

  Le problème n'existe pas sur les noyaux 2.6.
- La plupart des distributions Linux les plus courantes (Gentoo, SuSE, Mandriva, Debian, ...).
- Différentes versions de Solaris.
- FreeBSD (pilote de bande supporté à partir de la version 1.30 – allez voir les considérations **importantes** dans la section Configuration des lecteurs de bandes sur FreeBSD du chapitre Test des Bandes de ce manuel.)
- Windows (Win98/Me, WinNT/2K/XP) clients binaires (**File Daemon**).
- MacOS X/Darwin (voir http://fink.sourceforge.net/ pour obtenir les paquets)
- OpenBSD Client (**File Daemon**).
- Irix Client (**File Daemon**).
- Tru64
- **Bacula** est réputé fonctionner sur d'autres systèmes (AIX, BSDI, HPUX, ...) mais nous ne les avons pas testé personnellement.
- RHat 7.2 AS2, AS3, AS4, Fedora Core 2, SuSE SLES 7,8,9, Debian Woody et Sarge Linux sur S/390 et Linux sur zSeries.
- Voir le chapitre de Portage de la Documentation Pour Developpeurs pour les informations concernant le portage sur d'autres systèmes.

Lecteurs de bandes supportés

Même si votre lecteur est dans la liste ci dessous, vérifiez le Chapitre Test des Lecteurs Bandes de ce manuel pour les procédures que vous pouvez utiliser pour vérifier si votre lecteur de bande fonctionnera avec Bacula. Si votre lecteur est en mode bloc fixe, il peut sembler travailler avec Bacula jusqu'à ce que vous essayiez de restaurer et que Bacula tente de se positionner sur la bande. Seuls la procédure ci-dessus et vos propres tests peuvent vous garantir un fonctionnement correct.
Il est très difficile de fournir une liste de lecteurs de bandes supportés, ou de lecteurs qui sont connus pour fonctionner avec Bacula en raison d'un retour limité de la part des usagers. (par conséquent, si vous utilisez Bacula sur un lecteur qui ne figure pas dans la liste, merci de nous le faire savoir). Selon les informations provenant de nos utilisateurs, les lecteurs suivants sont connus pour fonctionner avec Bacula. Un trait d'union dans une colonne signifie "inconnu" :

OS	Fabr.	Media	Modèle
-	ADIC	DLT	Adic Scalar 100 DLT
-	ADIC	DLT	Adic Fastor 22 DLT
-	-	DDS	Compaq DDS 2,3,4
-	Exabyte	-	Lecteurs Exabyte de moins d
-	Exabyte	-	Exabyte VXA drives
-	HP	Travan 4	Colorado T4000S
-	HP	DLT	HP DLT drives
-	HP	LTO	HP LTO Ultrium drives
FreeBSD 4.10 RELEASE	HP	DAT	HP StorageWorks DAT72i
-	Overland	LTO	LoaderXpress LTO
-	Overland	-	Neo2000
-	OnStream	-	OnStream drives (see below)
-	Quantum	DLT	DLT-8000
Linux	Seagate	DDS-4	Scorpio 40
FreeBSD 4.9 STABLE	Seagate	DDS-4	STA2401LW
FreeBSD 5.2.1 pthreads patched RELEASE	Seagate	AIT-1	STA1701W
Linux	Sony	DDS-2,3,4	-
Linux	Tandberg	-	Tandbert MLR3
FreeBSD	Tandberg	-	Tandberg SLR6
Solaris	Tandberg	-	Tandberg SLR75
Linux Gentoo	ADIC	-	IBM Ultrium LTO I

Une liste des Librarires supportées figure dans le chapitre librairies (autochangers) de ce document, ou vous trou-

verez d'autres lecteurs de bandes qui fonctionne avec Bacula.

Lecteurs de bande non supportés

Auparavant les lecteurs de bandes OnStream IDE-SCSI ne fonctionnaient pas avec Bacula. A partir de la version 1.33 de Bacula et de la version 0.9.14 du pilote noyau ou supérieur,ce lecteur est supporté. Consultez le chapitre de test car vous devez le configurer pour fonctionner en mode blocs de taille fixe.
Les lecteurs QIC sont connus pour avoir nombre de particularités (taille de blocs fixe, et un EOF plutôt que deux pour terminer la bande). En conséquence, vous devrez être particulièrement attentif à sa configuration pour le faire fonctionner avec Bacula.

A l'attention des utilisateurs de FreeBSD !!!

A moins que vous n'ayez appliqué un correctif sur la bibliothèque pthreads de votre systèmes FreeBSD, vous perdrez des données quand Bacula aura rempli une bande et passera à la suivante. La raison en est que les bibliothèques pthreads sans correctifs échouent à retourner un état d'alerte à Bacula signalant que la fin de bande est proche. Consultez le chapitre test des lecteurs de bandes de ce manuel pour d'importantes informations concernant la configuration de votre lecteur de bande pour qu'il soit compatible avec **Bacula**.

Autochangeurs supportés

Pour des informations sur les libraries (autochangeurs) supportées, allez voir la section Libraries supportées du chapitre Libraries de ce manuel.

Spécifications des cartouches

Nous ne pouvons vraiment pas vous dire quel lecteur acheter pour utiliser Bacula. Cependant, nous pouvons recommander d'éviter les lecteurs DDS. La technologie est plutôt ancienne et ces lecteurs nécessitent de fréquents nettoyages. Les lecteurs DLT sont généralement bien meilleurs (technologie plus récente) et ne requièrent pas autant d'opérations de nettoyage.
Ci-dessous, vous trouverez une table de spécifications de cartouches DLT et LTO qui vous permettra de vous faire une idée de la capacité et de la rapidité des lecteurs et cartouches modernes. Les capacités reportées ici sont les natives, sans compression. Tous les lecteurs modernes pratiquent la compression matérielle, et les fabricants affichent souvent une capacité compressée avec un ratio de 2:1. Le facteur

de compression réel dépend principalement des données sauvegardées, mais je pense qu'un ratio 1,5:1 est beaucoup plus raisonnable (autrement dit, multipliez la valeur de la table par 1,5 pour obtenir une estimation grossière de la capacité compressée). Les taux de transfert sont arrondis au GB/hr le plus proche. Les valeurs sont fournies par les divers fabricants.

Le type de media est la désignation du fabricant, vous n'êtes pas obligé de l'utiliser (mais vous devriez...) dans vos ressources de configuration Bacula.

Type de media	Type de lecteur	Capacité des media	Taux de transfert
DDS-1	DAT	2 GB	?? GB/hr
DDS-2	DAT	4 GB	?? GB/hr
DDS-3	DAT	12 GB	5.4 GB/hr
Travan 40	Travan	20 GB	?? GB/hr
DDS-4	DAT	20 GB	11 GB/hr
VXA-1	Exabyte	33 GB	11 GB/hr
DAT-72	DAT	36 GB	13 GB/hr
DLT IV	DLT8000	40 GB	22 GB/hr
VXA-2	Exabyte	80 GB	22 GB/hr
Half-high Ultrum 1	LTO 1	100 GB	27 GB/hr
Ultrium 1	LTO 1	100 GB	54 GB/hr
Super DLT 1	SDLT 220	110 GB	40 GB/hr
VXA-3	Exabyte	160 GB	43 GB/hr
Super DLT I	SDLT 320	160 GB	58 GB/hr
Ultrium 2	LTO 2	200 GB	108 GB/hr
Super DLT II	SDLT 600	300 GB	127 GB/hr
VXA-4	Exabyte	320 GB	86 GB/hr
Ultrium 3	LTO 3	400 GB	216 GB/hr

Démarrer avec Bacula

Si vous êtes comme moi, vous voulez faire fonctionner Bacula immédiatement pour en avoir un aperçu, puis, plus tard, vous reviendrez en arrière pour lire et connaître tous les détails. C'est exactement ce que ce chapitre se propose d'accomplir : vous faire avancer rapidement sans tous les détails. Si vous voulez sauter la section sur les Pools, Volumes et Labels, vous pourrez toujours y revenir, mais s'il vous plaît, veuillez lire ce chapitre jusqu'à la fin, et en particulier suivre les instructions pour tester votre lecteur de bandes.
Nous supposons que vous êtes parvenus à construire et installer Bacula, sinon, vous devriez d'abord jeter un oeil aux Prérequis (système) puis au chapitre Compiler et installer Bacula de ce manuel.

Comprendre les Jobs et Schedules

Afin de rendre Bacula aussi flexible que possible, les directives lui sont données en plusieurs endroits. L'instruction principale est la ressource Job, qui définit un job. Un job de type sauvegarde consiste en général en un FileSet, un client, un Schedule pour un ou plusieurs niveaux ou horaires de sauvegardes, un Pool, ainsi que des instructions additionnelles. Un autre point de vue est de considérer le FileSet comme "Que sauvegarder ?", le Client comme "Qui sauvegarder ?", le Schedule comme "Quand sauvegarder ?" et le Pool comme "Où sauvegarder ?" (c'est à dire, "Sur quel volume ?)
Typiquement, une combinaison FileSet/Client aura un job correspondant. La plupart des directives, telles que les FileSets, Pools, Schedules, peuvent être mélangées et partagées entre les jobs. Ainsi, vous pouvez avoir deux définitions (ressources) de jobs sauvegardant différents serveurs et utilisant les mêmes Schedule, FileSet (sauvegardant donc les mêmes répertoires sur les deux machines) et peut-être même les mêmes Pools. Le Schedule définira quel type de sauvegarde sera exécuté et à quel moment (par exemple les Full le mercredi, les incrémentales le reste de la semaine), et lorsque plus d'un job utilise le même Schedule la priorité du job détermine lequel démarre en premier. Si vous avez de nombreux jobs, vous devriez utiliser la directive JobDefs, qui vous permet de définir des paramètres par défaut pour vos jobs, qui peuvent être changés au sein de la ressource Job, mais qui vous évitent de réécrire les mêmes paramètres pour chaque job. En plus des FileSets que vous voulez sauvegarder, vous devriez aussi avoir un job qui sauvegarde votre catalogue.
Enfin, sachez qu'en plus des jobs de type Backup, vous pouvez aussi utiliser des jobs de type restore, verify, admin, qui ont chacun des exigences différentes.

Comprendre les Pools, Volumes et Labels

Si vous avez utilisé un programme tel que **tar** pour sauvegarder votre système, les notions de Pools, Volumes et labels peuvent vous sembler un peu confuses au premier abord. Un Volume est un simple support physique (cartouche, ou simple fichier) sur lequel Bacula écrit vos données de sauvegarde. Les Pools regroupent les Volumes de sorte qu'une sauvegarde n'est pas restreinte à la capacité d'un unique Volume. Par conséquent, plutôt que de nommer explicitement les Volumes dans votre Job, vous spécifiez un Pool, et Bacula se chargera de sélectionner le prochain Volume utilisable du Pool et vous demandera de le monter.

Bien que les options de base soient spécifiées dans la ressource Pool du fichier de configuration du Director, le Pool **réel** est géré par le Catalogue Bacula. Il contient les informations de la ressource Pool (bacula-dir.conf) ainsi que les informations concernant tous les Volumes qui ont été ajoutés au Pool. L'ajout de Volumes se fait en principe manuellement depuis la Console grâce à la commande **label**.

Pour chaque Volume, Bacula gère une quantité d'informations considérable telles que les première et dernière date et heure d'écriture, le nombre de fichiers sur le Volume, le nombre de bytes sur le Volume, le nombre de montages, etc.

Pour que Bacula puisse lire ou écrire sur un Volume physique, celui-ci doit avoir reçu un étiquettage logiciel afin que Bacula soit assuré que le bon Volume est monté. Ceci s'effectue en principe manuellement depuis la Console grâce à la commande **label**.

Les étapes de création de Pool, ajout de Volumes à ce Pool, et écriture d'étiquettes logicielles sur les Volumes, peuvent sembler pénibles au premier abord, mais en fait, elles sont tout à fait simples à franchir, et elles vous permettent d'utiliser plusieurs Volumes (plutôt que d'être limité à la capacité d'un seul). Les Pools vous procurent aussi une flexibilité importante pour votre politique de sauvegarde. Par exemple, vous pouvez avoir un Pool de Volumes "Daily" pour vos sauvegardes Incrémentales et un Pool de Volumes "Weekly" pour vos sauvegardes complètes (Full). En spécifiant le Pool approprié dans les Jobs de sauvegarde quotidiens et hebdomadaires, vous garantissez qu'aucun Job quotidien n'écrira jamais sur un Volume du Pool réservé aux sauvegardes hebdomadaire et vice versa, et Bacula vous dira quelle cartouche est requise, et quand.

Pour plus de détails concernant les Pools, consultez la section Ressource Pool du chapitre sur la configuration du Director, ou poursuivez votre lecture, nous reviendrons plus tard sur ce sujet.

Paramétrage des fichiers de configuration de Bacula

Après avoir exécuté la commande **./configure** *ad hoc*, **make** et **make install**, si c'est la première fois que vous exécutez Bacula, vous devez créer des fichiers de configuration valides pour le Director, le File Daemon, le Storage Daemon et le programme Console. Si vous avez suivi nos recommandations, des fichiers de configuration par défaut ainsi que les binaires des *daemons* seront situés dans votre répertoire d'installation. Dans tous les cas les binaires se trouvent dans le répertoire que vous avez spécifié au niveau de l'option **--sbindir** de la commande **./configure**, et les fichiers de configuration se trouvent dans le répertoire que vous avez spécifié au niveau de l'option **--sysconfdir**.
Lors des paramétrages initiaux de Bacula, il vous faudra investir un peu de temps pour modifier les fichiers de configuration par défaut afin de les adapter à votre environnement. Ceci peut nécessiter de redémarrer Bacula plusieurs fois jusqu'à ce que tout fonctionne correctement. Ne cédez pas au désespoir. Une fois que vous aurez créé vos fichiers de configuration, vous n'aurez que rarement besoin de les changer et de redémarrer Bacula. Le gros du travail consistera à changer la cartouche quand elle est pleine.

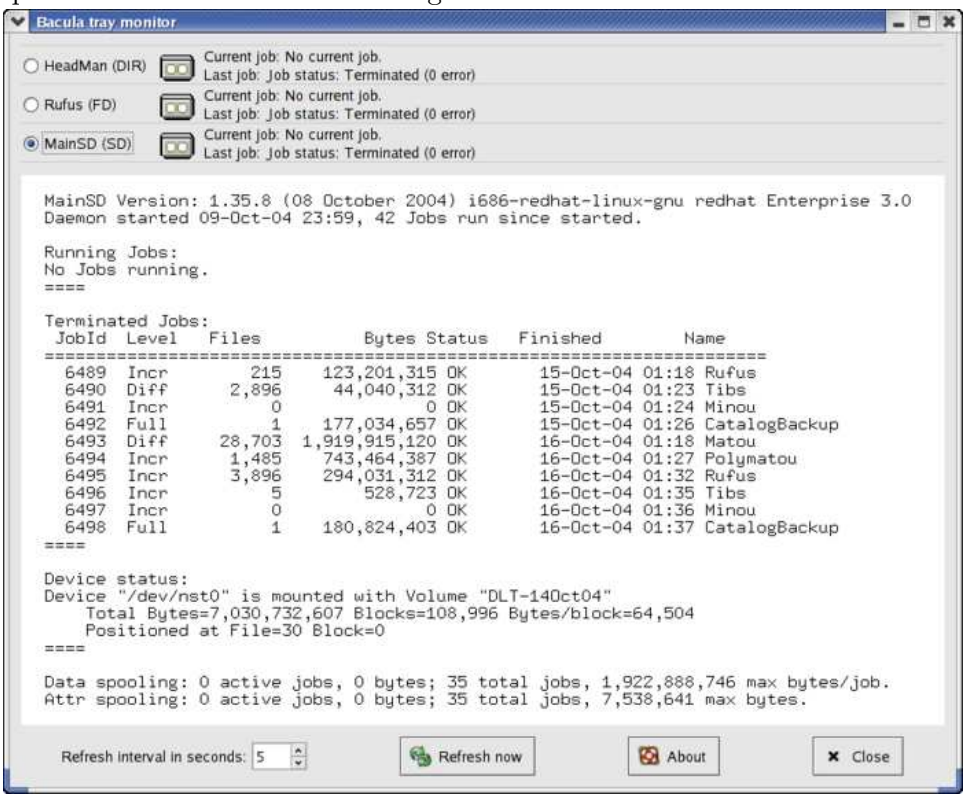
Configurer le programme Console

Le programme console est utilisé par l'administrateur pour interagir avec le Director et pour arrêter et démarrer manuellement des jobs, ou encore pour obtenir des informations sur les jobs en cours d'exécution ou programmés.
Le fichier de configuration de la Console se trouve dans le répertoire que vous avez spécifié au niveau de l'option **--sysconfdir** de la commande **./configure** et par défaut se nomme **console.conf**.
Si vous avez choisi de construire la Console GNOME avec l'option **--enable-gnome**, vous y trouverez également son fichier de configuration par défaut, nommé **gnome-console.conf**.
Il en va de même pour la console wxWidgets, qui est construite par l'option **--enable-wx-console**, et le nom du fichier de configuration par défaut est, dans ce cas, **wx-console.conf**.
Normalement, pour les nouveaux utilisateurs, aucune modification n'est requise pour ces fichiers. Les réglages par défaut sont raisonnables.

Configurer le programme Monitor

Le programme Monitor est typiquement une icône dans la barre des tâches. Cependant, lorsque l'icône est étendue en une fenètre, l'admi-

nistrateur ou l'utilisateur peut obtenir des informations concernant le Director ou l'état des sauvegardes sur la machine locale ou n'importe quelle autre *daemon* Bacula configuré.



L'image montre le tray-monitor configuré pour trois *daemons*. En cliquant sur les boutons radio dans le coin en haut à gauche de l'image, vous pouvez voir l'état de chacun des *daemons*. L'image montre l'état du Storage Daemon (MainSD) sélectionné.

Le fichier de configuration du Monitor se trouve dans le répertoire spécifié au niveau de l'option **--sysconfdir** de la commande **./configure** et se nomme par défaut **tray-monitor.conf**. En principe, pour les nouveaux utilisateurs, il suffit de changer les permissions de ce fichier pour permettre aux utilisateurs non-root d'exécuter le Monitor, en effet cette application doit être exécuté par le même utilisateur que l'environnement graphique (n'oubliez pas de donner aux non-root le droit d'exécuter **bacula-tray-monitor**). Ceci ne constitue pas une faille de sécurité tant que vous utilisez les réglages par défaut.

Configurer le File Daemon

Le File Daemon, est le programme qui s'exécute sur chaque machine cliente. A la demande du Director, il détermine les fichiers à sauvegarder et les expédie au Storage Daemon.

Le fichier de configuration du File Daemon se trouve dans le répertoire spécifié au niveau de l'option **--sysconfdir** de la commande **./configure** et se nomme par défaut **bacula-fd.conf**. Normalement, pour les nouveaux utilisateurs, aucune modification n'est requise pour ce fichier. Les réglages par défaut sont raisonnables. Cependant, si vous envisagez de sauvegarder plus d'une machine, il vous faudra installer le File Daemon avec un fichier de configuration spécifique sur chaque machine à sauvegarder. Les informations concernant chaque File Daemon doivent apparaître dans le fichier de configuration du Director.

Configurer le Director

Le director est le programme central qui contrôle tous les autres *daemons*. Il planifie et surveille les jobs à exécuter.
Le fichier de configuration du Director se trouve dans le répertoire spécifié au niveau de l'option **--sysconfdir** de la commande **./configure** et se nomme par défaut **bacula-dir.conf**.
En général, la seule modification nécessaire consiste à faire en sorte que la directive **Include** de la Ressource FileSet contienne au moins une ligne avec un nom de fichier ou de répertoire valide à sauvegarder.
Si vous ne possédez pas de lecteur DLT, vous voudrez probablement modifier la ressource Storage pour donner un nom plus représentatif de votre périphérique de stockage. Vous pouvez toujours utiliser les noms existants puisque vous êtes libre de les assigner arbitrairement, mais ils doivent s'accorder avec les noms correspondants dans le fichier de configuration du Storage Daemon.
Vous pouvez aussi changer l'adresse électronique pour les notifications vers votre propre adresse e-mail plutôt que vers celle de **root** (configuration par défaut).
Enfin, si vous avez plusieurs systèmes à sauvegarder, il vous faudra spécifier un File Daemon (ou client) pour chaque système sauvegardé, précisant ses nom, adresse et mot de passe. Nous estimons que baptiser vos *daemons* du nom de vos systèmes suffixés avec **-fd** aide beaucoup à corriger les erreurs. Ainsi, si votre système est **foobaz**, vous nommerez le *daemon* **foobaz-fd**. Pour le Director, vous pourriez utiliser **foobaz-dir**, et **foobaz-sd** pour le Storage Daemon. Chacun de vos composants de Bacula **doit** avoir un nom unique Si vous les nommez tous à l'identique, en plus de ne jamais savoir quel *daemon* envoie quel message, s'ils partagent le même répertoire de travail (working directory), les noms de fichiers temporaires des *daemons* ne seront pas uniques et vous aurez d'étranges erreurs.

Configurer le Storage Daemon

Le Storage Daemon est responsable, sur demande du Director, de la réception des données en provenance des File Daemons, et de leur écriture sur le medium de stockage, ou, dans le cas d'une restauration, de trouver les données pour les envoyer vers le File Daemon.
Le fichier de configuration du Storage Daemon se trouve dans le répertoire spécifié au niveau de l'option **--sysconfdir** de la commande **./configure** et se nomme par défaut **bacula-sd.conf**. Modifiez ce fichier pour accorder les noms de périphériques de stockage à ceux que vous possédez. Si le processus d'installation a convenablement détecté votre système, elles seront déjà correctement réglées. Ces ressources de stockage "Name" et "Media Type" doivent être les mêmes que leurs correspondantes du fichier de configuration du Director **bacula-dir.conf**. Si vous souhaitez sauvegarder vers un fichier plutôt que sur des bandes, la ressource Device doit pointer vers un répertoire où des fichiers seront créés en guise de Volumes lorque vous étiquetterez (label) vos Volumes.

Tester vos Fichiers de Configuration

Vous pouvez tester la validité syntaxique de vos fichiers de configuration, afficher tout message d'erreur et terminer. Par exemple, en supposant que vous avez installé vos binaires et fichiers de configuration dans le même répertoire,

```
cd <installation-directory>
./bacula-dir -t -c bacula-dir.conf
./bacula-fd -t -c bacula-fd.conf
./bacula-sd -t -c bacula-sd.conf
./bconsole -t -c bconsole.conf
./gnome-console -t -c gnome-console.conf
./wx-console -t -c wx-console.conf
su <normal user> -c "./bacula-tray-monitor -t -c tray-monitor.conf"
```

testera le fichier de configuration de chacun des principaux programmes. Si le fichier de configuration est correct, le programme se termine sans rien afficher. Veuillez noter que selon les options de configuration que vous avez choisies, il se peut qu'aucune des commandes ci-dessus ne soit valable sur votre système. Si vous avez installé les binaires dans les répertoires traditionnels d'Unix plutôt que dans un simple répertoire, il vous faudra modifier les commandes ci-dessus en conséquence (pas de "./" devant les commandes, et un chemin devant les fichiers de configuration).

Tester la compatibilité de Bacula avec votre lecteur de bandes

Avant de gaspiller votre temps avec Bacula pour finalement constater qu'il ne fonctionne pas avec votre lecteur de bandes, veuillez s'il vous plaît lire le chapitre btape – Tester votre lecteur de bandes de ce manuel. Si vous possédez un lecteur standard SCSI moderne sur un Linux ou un Solaris, fort probablement, il fonctionnera, mais mieux vaut tester que d'être déçu. Pour FreeBSD (et probablement les autres xBSD), la lecture du chapitre mentionné ci-dessus est un devoir. Pour FreeBSD, consultez aussi The FreeBSD Diary pour une description détaillée de la méthode pour faire fonctionner Bacula sur votre système. De plus, les utilisateurs de versions de FreeBSD antérieures à 4.9-STABLE datée du lundi 29 décembre 2003 15:18:01 UTC qui prévoient d'utiliser des lecteurs de bandes sont invités à lire le fichier **platforms/freebsd/pthreads-fix.txt** du répertoire principal de Bacula, qui contient d'importantes informations sur la compatibilité de Bacula avec leur système.

Débarrassez-vous du répertoire /lib/tls

La nouvelle librairie pthreads **/lib/tls** installée par défaut sur les systèmes Red Hat récents (kernels 2.4.x) est défectueuse. Vous devez la supprimer ou la renommer, puis rebooter votre système avant d'exécuter Bacula, faute de quoi, après environ une semaine de fonctionnement, Bacula se bloquera pour de longues périodes, voire définitivement. Veuillez consulter le chapitre Systèmes supportés pour plus d'informations sur ce problème.
Ce problème n'existe plus avec les noyaux 2.6.

Exécuter Bacula

La partie la plus importante de l'exécution de Bacula est probablement la capacité de restaurer les fichiers. Si vous n'avez pas essayé de récupérer des fichiers au moins une fois, vous subirez une bien plus forte pression le jour où vous devrez réellement le faire, et serez enclin à commettre des erreurs que vous n'auriez pas commises si vous aviez déjà essayé.
Pour avoir rapidement une bonne idée de la façon d'utiliser Bacula, nous vous recommandons **fortement** de suivre les exemples du chapitre exécuter Bacula de ce manuel, où vous trouverez des informations détaillées sur l'exécution de Bacula.

Rotation des logs

Si vous utilisez le **bacula-dir.conf** par défaut ou une variante, vous constaterez qu'il récupère toutes les sorties de Bacula dans un fichier. Pour éviter que ce fichier ne croisse sans limites, nous vous recommandons de copier le fichier **logrotate** depuis **scripts/logrotate** vers **/etc/logrotate.d/bacula**. Ainsi les fichiers de logs subiront une rotation mensuelle et seront conservés pour une durée maximum de cinq mois. Vous pouvez éditer ce fichier pour adapter la rotation à votre convenance.

Log Watch

Certains systèmes tels que RedHat et Fedora exécutent le programme logwatch chaque nuit pour analyser vos fichiers de log et vous envoyer un rapport par mail. Si vous souhaitez inclure la sortie de vos jobs Bacula dans ce rapport, veuillez regarder dans le répertoire **scripts/logwatch**. Le fichier **README** fournit une brève explication sur la faon d'installer le script, et quelle genre de résultats en attendre.

Reprise d'activité après un désastre (disaster recovery)

Si vous avez l'intention d'utiliser Bacula en tant qu'outil de disaster recovery plutôt que comme un simple programme pour restaurer les fichiers perdus, vous serez intéressé par le chapitre Plan de reprise d'activité avec Bacula de ce manuel.

De toute façon, vous êtes fortement invité à tester soigneusement la restauration de quelques fichiers que vous aurez préalablement sauvegardés, plutôt que d'attendre qu'un désastre ne frappe. Ainsi, vous serez préparé.

Installer Bacula

Prérequis

En général, il vous faudra les sources de la version courante de Bacula, et si vous souhaitez exécuter un client Windows, vous aurez besoin de la version binaire du client Bacula pour Windows. Par ailleurs, Bacula a besoin de certains paquetages externes (tels **SQLite**, **MySQL** ou **PostgreSQL**) pour compiler correctement en accord avec les options que vous aurez choisies. Pour vous simplifier la tâche, nous avons combiné plusieurs de ces programmes dans deux paquetages **depkgs** (paquetages de dépendances). Ceci peut vous simplifier la vie en vous fournissant tous les paquets nécessaires plutôt que de vous contraindre à les trouver sur la Toile, les charger et installer.

Distribution des fichiers source

A partir de la version 1.38.0, le code source est éclaté en quatre fichiers tar correspondant à quatre modules différents dans le CVS Bacula. Ces fichiers sont :

bacula-1.38.0.tar.gz Il s'agit de la distribution primaire de Bacula. Pour chaque nouvelle version, le numéro de version (ici, 1.38.0) sera mise à jour.

bacula-docs-1.38.0.tar.gz Ce fichier contient une copie du répertoire docs, avec les documents pré-construits : Répertoire html anglais, fichier html unique et fichier pdf. Les traductions allemande et franaise sont en cours mais ne sont pas pré-construites.

bacula-gui-1.38.0.tar.gz Ce fichier contient les programmes graphique en dehors du coeur de l'application. Actuellement, il contient bacula-web, un programme PHP pour produire une vue d'ensemble des statuts de vos jobs Bacula consultable dans un navigateur ; et bimagemgr, un programme qui permet de graver des images de CDROMS depuis un navigateur avec les volumes Bacula.

bacula-rescue-1.8.1.tar.gz Ce fichier contient le code du CDROM de secours Bacula. Notez que le numéro de version de ce paquetage n'est pas lié à celui de Bacula. En utilisant ce code, vous pouvez graver un CDROM contenant la configuration de votre système et une version statiquement liée du File Daemon. Ceci peut vous permettre de repartitionner et reformater aisément vos disques durs et de recharger votre système avec Bacula en cas de défaillance du disque dur.

Mettre Bacula à jour

Si vous faites une mise à jour de Bacula, vous devriez d'abord lire attentivement les ReleaseNotes de toutes les versions entre votre version installée et celle vers laquelle vous souhaitez mettre à jour. Si la base de données du catalogue a été mise à jour, vous devrez soit réinitialiser votre base de données et repartir de zéro, soit en sauvegarder une copie au format ASCII avant de procéder à sa mise à jour. Ceci est normalement fait lorsque Bacula est compilé et installé par :

```
cd <installed-scripts-dir> (default /etc/bacula)
./update_bacula_tables
```

Ce script de mise à jour peut aussi être trouvé dans le répertoire src/cats des sources de Bacula.
S'il y a eu plusieurs mises à jour de la base de données entre votre version et celle vers laquelle vous souhaitez évoluer, il faudra appliquer chaque script de mise à jour de base de données. Vous pouvez trouver tous les anciens scripts de mise à jour dans le répertoire **upgradedb** des sources de Bacula. Il vous faudra éditer ces scripts pour qu'ils correspondent à votre configuration. Le script final, s'il y en a un, sera dans le répertoire **src/cats** comme indiqué dans la ReleaseNote.
Si vous migrez d'une version majeure vers une autre, vous devrez remplacer tous vos composants (*daemons*) en même temps car, généralement, le protocole inter-*daemons* aura changé. Par contre, entre deux versions mineures d'une même majeure (par exemple les versions 1.32.x), à moins d'un bug, le protocole inter-*daemons* ne changera pas. Si cela vous semble confus, lisez simplement les ReleaseNotes très attentivement, elles signaleront si les *daemons* doivent être mis à jour simultanément.
Enfin, notez qu'il n'est généralement pas nécessaire d'utiliser **make uninstall** avant de procéder à une mise à jour. En fait, si vous le faites vous effacerez probablement vos fichiers de configuration, ce qui pourrait être désastreux. La procédure normale de mise à jour est simplement **make install**. En principe, aucun de vos fichiers .conf ou .sql ne sera écrasé.
Pour plus d'informations sur les mises à jour, veuillez consulter la partie Upgrading Bacula Versions du chapitre Astuces de ce manuel

Paquetage de Dépendences

Comme nous l'évoquions plus haut, nous avons combiné une série de programmes dont Bacula peut avoir besoin dans les paquets **depkgs** et **depkgs1**. Vous pouvez, bien sur, obtenir les paquets les plus récents

directement des auteurs. Le fichier README dans chaque paquet indique où les trouver. Pourtant, il faut noter que nous avons testé la compatibilité des paquets contenus dans les fichiers depkgs avec Bacula.
Vous pouvez, bien sur, obtenir les dernieres versions de ces paquetages de leurs auteurs. Les références nécessaires figurent dans le README de chaque paquet. Quoi qu'il en soit, soyez conscient du fait que nous avons testé la compatibilité des paquetages des fichiers depkgs.
Typiquement, un paquetage de dépendances sera nommé **depkgs-ddMMMyy.tar.gz** et **depkgs1-ddMMMyy.tar.gz** où **dd** est le jour où n'ous l'avons publié, **MMM** l'abbréviation du mois et **yy** l'année. Par exemple: **depkgs-07Apr02.tar.gz**. Pour installer et construire ce paquetage (s'il est requis), vous devez:

1. Créer un répertoire **bacula**, dans lequel vous placerez les sources de Bacula et le paquetage de dépendances.
2. Désarchiver le **depkg** dans le répertoire **bacula**.
3. vous déplacer dans le répertoire obtenu: cd bacula/depkgs
4. exécuter make

La composition exacte des paquetages de dépendance est susceptible de changer de temps en temps, voici sa composition actuelle :

Paquets externes	**depkgs**	**depkgs1**	**depkgs-win32**
SQLite	X	-	-
mtx	X	-	-
readline	-	X	-
pthreads	-	-	X
zlib	-	-	X
wxWidgets	-	-	X

Notez que certains de ces paquets sont de taille respectable, si bien que l'étape de compilation peut prendre un certain temps. Les instructions ci-dessous construiront tous les paquets contenus dans le répertoire. Cependant, la compilation de Bacula, ne prendra que les morceaux dont Bacula a effectivement besoin.
Une alternative consiste à ne construire que les paquets nécessaires. Par exemple,

```
cd bacula/depkgs
make sqlite
```

configurera et construira SQLite et seulement SQLite.
Vous devriez construire les paquets requis parmi **depkgs** et/ou **depkgs1** avant de configurer et compiler Bacula car Bacula en aura

besoin dès la compilation.
Même si vous n'utilisez pas SQLite, vous pourriez trouver le paquet **depkgs** pratique pour construire **mtx** car le programme **tapeinfo** qui vient avec peut souvent vous fournir de précieuses informations sur vos lecteurs de bandes SCSI (e.g. compression, taille min/max des blocks,...).
Le paquet **depkgs-win32** contient le code source pour les librairies pthreads et zlib utilisées par le client Win32 natif. Vous n'en aurez besoin que si vous prévoyez de construire le client Win32 depuis les sources.

Systèmes Supportés

Veuillez consulter la section Systèmes supportés du chapitre Démarrer avec Bacula de ce manuel.

Construire Bacula à partir des sources

L'installation basique est plutôt simple.

1. Installez et construisez chaque **depkgs** comme indiqué plus haut.
2. Configurez et installez MySQL ou PostgreSQL (si vous le souhaitez): Installer et configurer MySQL Phase I ou Installer et configurer PostgreSQL Phase I. Si vous installez depuis des rpms, et utilisez MySQL, veillez à installer **mysql-devel**, afin que les fichiers d'en-têtes de MySQL soient disponibles pour la compilation de Bacula. De plus, la librairie client MySQL requièrt la librairie de compression gzip **libz.a** ou **libz.so**. Ces librairies sont dans le paquet **libz-devel**. Sur Debian, vous devrez charger le paquet **zlib1g-dev**. Si vous n'utilisez ni rpms, ni debs, il vous faudra trouver le paquetage adapté à votre système.
   Notez que si vous avez dejà MySQL ou PostgreSQL sur votre système vous pouvez sauter cette phase pourvu que vous ayez construit "the thread safe libraries" et que vous ayez déjà installé les rpms additionnels sus-mentionnés.
3. En alternative à MySQL et PostgreSQL, configurez et installez SQLite, qui fait partie du paquetage **depkgs**. Installer et configurer SQLite.
4. Désarchivez les sources de Bacula, de préférence dans le répertoire **bacula** évoqué ci-dessus.
5. Déplacez-vous dans ce répertoire.
6. Exécutez ./configure (avec les options appropriées comme décrit ci-dessus)

7. Examinez très attentivement la sortie de ./configure, particulièrement les répertoires d'installation des binaires et des fichiers de configuration. La sortie de ./configure est stockée dans le fichier **config.out** et peut être affichée à volonté sans relancer ./configure par la commande **cat config.out**.
8. Vous pouvez relancer ./configure avec des options différentes après une première exécution, cela ne pose aucun problème, mais vous devriez d'abord exécuter:

   ```
   make distclean
   ```

   afin d'être certain de repartir de zéro et d'éviter d'avoir un mélange avec vos premières options. C'est nécessaire parce que ./configure met en cache une bonne partie des informations. **make distclean** est aussi recommandé si vous déplacez vos fichiers source d'une machine à une autre. Si **make distclean** échoue, ignorez-le et continuez.
9. make
   Si vous obtenez des erreurs durant le *linking* dans le répertoire du Storage Daemon (/etc/stored), c'est probablement parce que vous avez chargé la librairie statique sur votre système. J'ai remarqué ce problème sur un Solaris. Pour le corriger, assurez-vous de ne pas avoir ajouté l'option **--enable-static-tools** à la commande **./configure**.
   Si vous ignorez cette étape (**make**) et poursuivez immédiatement avec **make install**, vous commettez deux erreurs sérieuses : d'abord, votre installation va échouer car Bacula a besoin d'un **make** avant un **make install** ; ensuite, vous vous privez de la possibilité de vous assurer qu'il n'y a aucune erreur avant de commencer écrire les fichiers dans vos répertoires système.
10. make install
11. Si vous êtes un nouvel utilisateur de Bacula, nous vous recommandons **fortement** de sauter l'étape suivante et d'utiliser le fichier de configuration par défaut, puis d'exécuter le jeu d'exemples du prochain chapitre avant de revenir modifier vos fichier de configuration pour qu'ils satisfassent vos besoins.
12. Modifiez les fichiers de configuration de chacun des trois *daemons* (Directory, File, Storage) et celui de la Console. Pour plus de détails, consultez le chapitre Fichiers de Configuration de Bacula Nous vous recommandons de commencer par modifier les fichiers de configuration fournis par défaut, en faisant les changements minima indispensables. Vous pourrez procéder à une adaptation complète une fois que Bacula fonctionnera correctement. Veuillez prendre garde à modifier les mots de passe qui sont générés

aléatoirement, ainsi que les noms car ils doivent s'accorder entre les fichiers de configuration pour des raisons de sécurité.

13. Créez la base de données Bacula MySQL et ses tables (si vous utilisez MySQL) Installer et configurer MySQL Phase II ou créez la base de données Bacula PostgreSQL et ses tables Installer et configurer PostgreSQL Phase II (si vous utilisez PostgreSQL) ou encore Installer et configurer SQLite Phase II (si vous utilisez SQLite)
14. Démarrez Bacula (**./bacula start**) Notez: Le prochain chapitre expose ces étapes en détail.
15. Lancez la Console pour communiquer avec Bacula.
16. Pour les deux éléments précédents, veuillez suivre les instructions du chapitre Exécuter Bacula où vous ferez une simple sauvegarde et une restauration. Faites ceci avant de faire de lourdes modifications aux fichiers de configuration, ainsi vous serez certain que Bacula fonctionne, et il vous sera plus familier. Après quoi il vous sera plus facile de changer les fichiers de configuration.
17. Si après l'installation de Bacula, vous décidez de le déplacer, c'est à dire de l'installer dans un jeu de répertoires différents, procédez comme suit :

```
make uninstall
make distclean
./configure (vos-nouvelles-options)
make
make install
```

Si tout se passe bien, **./configure** déterminera correctement votre système et configurera correctement le code source. Actuellement, FreeBSD, Linux (RedHat), et Solaris sont supportés. Des utilisateurs rapportent que le client Bacula fonctionne sur MacOS X 10.3 tant que le support readline est désactivé.

Si vous installez Bacula sur plusieurs systèmes identiques, vous pouvez simplement transférer le répertoire des sources vers ces autres systèmes et faire un "make install". Cependant s'il y a des différences dans les librairies, ou les versions de systèmes, ou si vous voulez installer sur un système différent, vous devriez recommencer à partir de l'archive tar compressée originale. Si vous transférez un répertoire de sources où vous avez déjà exécuté la commande ./configure, vous DEVEZ faire:

```
make distclean
```

avant d'exécuter à nouveau ./configure. Ceci est rendu nécessaire par l'outil GNU autoconf qui met la configuration en cache, de sorte que si vous réutilisez la configuration d'une machine Linux sur un Solaris,

vous pouvez être certain que votre compilation échouera. Pour l'éviter, comme mentionné plus haut, recommencez depuis l'archive tar, ou faites un "make distclean".
En général, vous voudrez probablement sophistiquer votre **configure** pour vous assurer que tous les modules que vous souhaitez soient construits et que tout soit placé dans les bons répertoires.
Par exemple, sur RedHat, on pourrait utiliser ceci:

```
CFLAGS="-g -Wall" \
  ./configure \
    --sbindir=$HOME/bacula/bin \
    --sysconfdir=$HOME/bacula/bin \
    --with-pid-dir=$HOME/bacula/bin/working \
    --with-subsys-dir=$HOME/bacula/bin/working \
    --with-mysql=$HOME/mysql \
    --with-working-dir=$HOME/bacula/bin/working \
    --with-dump-email=$USER
```

Notez: l'avantage de cette configuration pour commencer, est que tout sera mis dans un seul répertoire, que vous pourrez ensuite supprimer une fois que vous aurez exécuté les exemples du prochain chapitre, et appris comment fonctionne Bacula. De plus, ceci peut être installé et exécuté sans être root.
Pour le confort des développeurs, j'ai ajouté un script **defaultconfig** au répertoire **examples**. Il contient les réglages que vous devriez normalement utiliser, et chaque développeur/utilisateur devrait le modifier pour l'accorder à ses besoins. Vous trouverez d'autres exemples dans ce répertoire.
Les options **--enable-conio** ou **--enable-readline** sont utiles car elles fournissent un historique de lignes de commandes et des capacités d'édition à la Console. Si vous avez inclus l'une ou l'autre option, l'un des deux paquets **termcap** ou **ncurses** sera nécessaire pour compiler. Sur certains systèmes, tels que SUSE, la librairie termcap n'est pas dans le répertoire standard des librairies par conséquent, l'option devrait être désactivée ou vous aurez un message tel que:

```
/usr/lib/gcc-lib/i586-suse-linux/3.3.1/.../ld:
cannot find -ltermcap
collect2: ld returned 1 exit status
```

lors de la compilation de la Console Bacula. Dans ce cas, il vous faudra placer la variable d'environnement **LDFLAGS** avant de compiler.

```
export LDFLAGS="-L/usr/lib/termcap"
```

Les mêmes contraintes de librairies s'appliquent si vous souhaitez utiliser les sous-programmes readlines pour l'édition des lignes de commande et l'historique, ou si vous utilisez une librairie MySQL qui requiert le chiffrement. Dans ce dernier cas, vous pouvez exporter les

librairies additionnelles comme indiqué ci-dessus ou, alternativement, les inclure directement en paramètres de la commande ./configure comme ci-dessous :

```
LDFLAGS="-lssl -lcyrpto" \
   ./configure \
         <vos-options>
```

Veuillez noter que sur certains systèmes tels que Mandriva, readline tend à "avaler" l'invite de commandes, ce qui le rend totalement inutile. Si cela vous arrive, utilisez l'option "disable", ou si vous utilisez une version postérieure à 1.33 essayez **--enable-conio** pour utiliser une alternative à readline intégrée. Il vous faudra tout de même termcap ou ncurses, mais il est peu probable que le paquetage **conio** gobe vos invites de commandes.
Readline n'est plus supporté depuis la version 1.34. Le code reste disponible, et si des utilisateurs soumettent des patches, je serai heureux de les appliquer. Cependant, étant donné que chaque version de readline semble incompatible avec les précédentes, et qu'il y a des différences significatives entre les systèmes, je ne puis plus me permettre de le supporter.

Quelle base de données utiliser ?

Avant de construire Bacula, vous devez décider si vous voulez utiliser SQLite, MySQL ou PostgreSQL. Si vous n'avez pas déjà MySQL ou PostgreSQL sur votre machine, nous vous recommandons de démarrer avec SQLite. Ceci vous facilitera beaucoup l'installation car SQLite est compilé dans Bacula et ne requiert aucune administration. SQLite fonctionne bien et sied bien aux petites et moyennes configurations (maximum 10-20 machines). Cependant, il nous faut signaler que plusieurs utilisateurs ont subi des corruptions inexpliquées de leur catalogue SQLite. C'est pourquoi nous recommandons de choisir MySQL ou PostgreSQL pour une utilisation en production.
Si vous souhaitez utiliser MySQL pour votre catalogue Bacula, consultez le chapitre Installer et Configurer MySQL de ce manuel. Vous devrez installer MySQL avant de poursuivre avec la configuration de Bacula. MySQL est une base de données de haute qualité très efficace et qui convient pour des configurations de toutes tailles. MySQL est légèrement plus complexe à installer et administrer que SQLite en raison de ses nombreuses fonctions sophistiquées telles que userids et mots de passe. MySQL fonctionne en tant que processus distinct, est réellement une solution professionnelle et peut prendre en charge des bases de données de dimension quelconque.

Si vous souhaitez utiliser PostgreSQL pour votre catalogue Bacula, consultez le chapitre Installer et Configurer PostgreSQL de ce manuel. Vous devrez installer PostgreSQL avant de poursuivre avec la configuration de Bacula. PostgreSQL est très similaire à MySQL bien que tendant à être un peu plus conforme à SQL92. PostgreSQL possède beaucoup plus de fonctions avancées telles que les transactions, les procédures stockées, etc. PostgreSQL requiert une certaine connaissance pour son installation et sa maintenance.
Si vous souhaitez utiliser SQLite pour votre catalogue Bacula, consultez le chapitre Installer et Configurer SQLite de ce manuel.

Démarrage rapide

Il y a de nombreuses options et d'importantes considérations données ci-dessous que vous pouvez passer pour le moment si vous n'avez eu aucun problème lors de la compilation de Bacula avec une configuration simplifiée comme celles montrées plus haut.
Si le processus ./configure ne parvient pas à trouver les librairies spécifiques (par exemple libintl), assurez vous que le paquetage approprié est installé sur votre système. S'il est installé dans un répertoire non standard (au moins pour Bacula), il existe dans la plupart des cas une option parmi celles énumérées ci-dessous (ou avec "./configure --help") qui vous permettra de spécifier un répertoire de recherche. D'autres options vous permettent de désactiver certaines fonctionnalités (par exemple --disable-nls).
Si vous souhaitez vous jeter à l'eau, nous vous conseillons de passer directement au chapitre suivant, et d'exécuter le jeu d'exemples. Il vous apprendra beaucoup sur Bacula, et un Bacula de test peut être installé dans un unique répertoire (pour une destruction aisée) et exécuté sans être root. Revenez lire les détails de ce chapitre si vous avez un quelconque problème avec les exemples, ou lorsque vous voudrez effectuer une installation réelle.
TAQUET MISE A JOUR

Options de la commande configure

Les options en ligne de commande suivantes sont disponibles pour **configure** afin d'adapter votre installation à vos besoins.

--sysbindir=<binary-path> Définit l'emplacement des binaires Bacula.

--sysconfdir=<config-path> Définit l'emplacement des fichiers de configuration de Bacula.

--mandir=<path> Par défaut, Bacula installe une simple page de manuel dans /usr/share/man. Si vous voulez qu'elle soit installée ailleurs, utilisez cette options pour spécifier le chemin voulu. Notez que les principaux documents Bacula en HTML et PDF sont dans une archive tar distincte des sources de distribution de Bacula.

--datadir=<path> Si vous traduisez Bacula ou des parties de Bacula dans une autre langue, vous pouvez spécifier l'emplacement des fichiers .po avec l'option **--datadir**. Vous devez installer manuellement tout fichier .po qui n'est pas (encore) installé automatiquement.

--enable-smartalloc Permet l'inclusion du code Smartalloc de détection de tampons orphelins (NDT : orphaned buffer). Cette option est vivement recommandée. Nous n'avons jamais compilé sans elle, aussi vous pourriez subir des désagréments si vous ne l'activez pas. Dans ce cas, réactivez simplement cette option. Nous la recommandons car elle aide à détecter les fuites de mémoire. Ce paramètre est utilisé lors de la compilation de Bacula.

--enable-gnome Si vous avez installé GNOME sur votre ordinateur, vous devez spécifier cette option pour utiliser la Console graphique GNOME. Vous trouverez les binaires dans le répertoire **src/gnome-console**.

--enable-wx-console Si vous avez installé wxWidgets sur votre ordinateur, vous devez spécifier cette option pour utiliser la Console graphique wx-console. Vous trouverez les binaires dans le répertoire **src/wx-console**. Ceci peut être utile aux utilisateurs qui veulent une Console graphique, mais ne souhaitent pas installer Gnome, car wxWidgets peut fonctionner avec les librairies GTK+, Motif ou même X11.

--enable-tray-monitor Si vous avez installé GTK sur votre ordinateur et utilisez un gestionnaire de fenêtre compatible avec le système de notification standard FreeDesktop (tels KDE et GNOME), vous pouvez utiliser une interface graphique pour surveiller les *daemons* Bacula en activant cette option. Les binaires seront placés dans le répertoire **src/tray-monitor**.

--enable-static-tools Avec cette option, les utilitaires relatifs au Storage Daemon (**bls**, **bextract**, et **bscan**) seront liés statiquement, ce qui vous permet de les utiliser même si les librairies partagées ne sont pas chargées. Si vous avez des difficultés de type "linking" à la compilation du répertoire **src/stored**, assurez-vous d'avoir désactivé cette option, en ajoutant éventuellement **--disable-static-tools**.

--enable-static-fd Avec cette option, la compilation produira un **static-bacula-fd** en plus du File Daemon standard. Cette version qui inclut les librairies statiquement liées est requise pour la reconstruction complète d'une machine après un désastre. Cette option est largement surpassée par l'usage de **make static-bacula-fd** du répertoire **src/filed**. L'option **--enable-client-only** décrite plus loin est aussi intéressante pour compiler un simple client sans les autres parties du programme.

Pour lier un binaire statique, l'éditeur de liens a besoin des versions statiques de toutes les librairies utilisées, aussi les utilisateurs rencontrent fréquemment des erreurs d'édition de liens à l'utilisation de cette option. La première chose à faire est de s'assurer d'avoir la librairie glibc statiquement liée sur votre système. Ensuite, il faut s'assurer de ne pas utiliser les options **--openssl** ou **--with-python** de la commande configure, car elle requierent des librairies supplémentaires. Vous devriez pouvoir activer ces options, mais il vous faudra charger les librairies statiques additionnelles correspondantes.

--enable-static-sd Avec cette option, la compilation produira un **static-bacula-sd** en plus du Storage Daemon standard. Cette version qui inclut les librairies statiquement liées peut se révéler utile pour la reconstruction complète d'une machine après un désastre.

Pour lier un binaire statique, l'éditeur de liens a besoin des versions statiques de toutes les librairies utilisées, aussi les utilisateurs rencontrent fréquemment des erreurs d'édition de liens à l'utilisation de cette option. La première chose à faire est de s'assurer d'avoir la librairie glibc statiquement liée sur votre système. Ensuite, il faut s'assurer de ne pas utiliser les options **--openssl** ou **--with-python** de la commande configure, car elle requierent des librairies supplémentaires. Vous devriez pouvoir activer ces options, mais il vous faudra charger les librairies statiques additionnelles correspondantes.

--enable-static-dir Avec cette option, la compilation produira un **static-bacula-dir** en plus du Director Daemon standard. Cette version qui inclut les librairies statiquement liées peut se révéler utile pour la reconstruction complète d'une machine après un désastre.

Pour lier un binaire statique, l'éditeur de liens a besoin des versions statiques de toutes les librairies utilisées, aussi les utilisateurs rencontrent fréquemment des erreurs d'édition de liens à l'utilisation de cette option. La première chose à faire est de s'assurer d'avoir la librairie glibc statiquement liée sur votre système.

Ensuite, il faut s'assurer de ne pas utiliser les options **--openssl** ou **--with-python** de la commande configure, car elle requierent des librairies supplémentaires. Vous devriez pouvoir activer ces options, mais il vous faudra charger les librairies statiques additionnelles correspondantes.

--enable-static-cons Avec cette option, la compilation produira une **static-console** et une **static-gnome-console** en plus de la Console standard standard. Cette version qui inclut les librairies statiquement liées peut se révéler utile pour la reconstruction complète d'une machine après un désastre.

Pour lier un binaire statique, l'éditeur de liens a besoin des versions statiques de toutes les librairies utilisées, aussi les utilisateurs rencontrent fréquemment des erreurs d'édition de liens à l'utilisation de cette option. La première chose à faire est de s'assurer d'avoir la librairie glibc statiquement liée sur votre système. Ensuite, il faut s'assurer de ne pas utiliser les options **--openssl** ou **--with-python** de la commande configure, car elle requierent des librairies supplémentaires. Vous devriez pouvoir activer ces options, mais il vous faudra charger les librairies statiques additionnelles correspondantes.

--enable-client-only Avec cette option, la compilation produira seulement le File Daemon et les librairies qui lui sont nécessaires. Aucun des autres *daemons*, outils de stockage, ni la console ne sera compilé. De même, un **make install** installera seulement le File Daemon. Pour obtenir tous les *daemons*, vous devez la désactiver. Cette option facilite grandement la compilation sur les simples clients.

Pour lier un binaire statique, l'éditeur de liens a besoin des versions statiques de toutes les librairies utilisées, aussi les utilisateurs rencontrent fréquemment des erreurs d'édition de liens à l'utilisation de cette option. La première chose à faire est de s'assurer d'avoir la librairie glibc statiquement liée sur votre système. Ensuite, il faut s'assurer de ne pas utiliser les options **--openssl** ou **--with-python** de la commande configure, car elle requierent des librairies supplémentaires. Vous devriez pouvoir activer ces options, mais il vous faudra charger les librairies statiques additionnelles correspondantes.

--enable-largefile Cette option (activée par défaut) provoque la compilation de Bacula avec le support d'adressage de fichiers 64 bits s'il est disponible sur votre système. Ainsi Bacula peut lire et écrire des fichiers de plus de 2 GBytes. Vous pouvez désactiver cette option et revenir à un adressage de fichiers 32 bits en utilisant **--disable-largefile**.

--disable-nls Bacula utilise par défaut les librairies *GNU Native Language Support* (NLS). Sur certaines machines, ces librairies peuvent être inexistante, ou ne pas fonctionner correctement (particulièrement sur les implémentations non Linux). dans ce genre de situations, vous pouvez neutraliser l'utilisation de ces librairies avec l'option **--disable-nls**. Dans ce cas, Bacula reviendra à l'usage de l'anglais.

--with-sqlite=<sqlite-path> Cette option permet l'utilisation de la base de données SQLite versions 2.8.x. Il n'est, en principe, pas nécessaire de spécifier le chemin **sqlite-path** car Bacula recherche les composants requis dans les répertoires standards (**depkgs/sqlite**). voyez Installer et Configurer SQLite pour plus de détails.

Voyez aussi la note ci-dessous, après le paragraphe –with-postgreSQL

--with-sqlite3=<sqlite3-path> Cette option permet l'utilisation de la base de données SQLite versions 3.x. Il n'est, en principe, pas nécessaire de spécifier le chemin **sqlite3-path** car Bacula recherche les composants requis dans les répertoires standards (**depkgs/sqlite3**). voyez Installer et Configurer SQLite pour plus de détails.

Voyez aussi la note ci-dessous, après le paragraphe –with-postgreSQL

--with-mysql=<mysql-path> Cette option permet la compilation des services de Catalogue de Bacula. Elle implique que MySQL tourne déjà sur votre système, et qu'il soit installé dans le chemin **mysql-path** que vous avez spécifié. Si cette option est absente, Bacula sera compilé automatiquement avec le code de la base Bacula interne. Nous recommandons d'utiliser cette option si possible. Si vous souhaitez utilisez cette option, veuillez procéder à l'installation de MySQL (Installer and Configurer MySQL) avant de procéder à la configuration.

Voyez aussi la note ci-dessous, après le paragraphe –with-postgreSQL

--with-postgresql=<postgresql-path> Cette option déclare un chemin explicite pour les librairies PostgreSQL si Bacula ne les trouve pas dans le répertoire par défaut.

Notez que pour que Bacula soit configuré correctement, vous devez spécifier l'une des quatre options de bases de données supportées : --with-sqlite, --with-sqlite3, --with-mysql, ou --with-postgresql, faute de quoi ./configure échouera.

--with-openssl=<path> Cette option est requise si vous souhaitez activer TLS (ssl) dans Bacula. Normalement, la spécification du

chemin **path** n'est pas nécessaire car le processus de configuration recherche les librairies OpenSSL dans les emplacements standard du système. L'activation d'OpenSSL dans Bacula permet des communications sécurisées entre les *daemons*. Pour plus d'informations sur l'usage de TLS, consultez le chapitre Bacula TLS de ce manuel.

--with-python=<path> Cette option active le support Python dans Bacula. Si le chemin n'est pas spécifié, le processus de configuration recherchera les librairies Python dans leurs emplacements standard. S'il ne peut trouver les librairies , il vous faudra fournir le chemin vers votre répertoire de librairies Python. Voyez le chapitre Python pour plus de détails sur l'utilisation de scripts Python.

--with-libintl-prefix=<DIR> Cette option peut être utilisée pour indiquer à Bacula de rechercher dans DIR/include et DIR/lib les fichiers d'en tête libintl et les librairies requises pour Native Language Support (NLS).

--enable-conio Cette option permet la compilation d'une petite et légère routine en alternative à readline, beaucoup plus facile à configurer, même si elle nécessite aussi les librairies termcap ou ncurses.

--with-readline=<readline-path> Spécifie l'emplacement de **readline**. En principe, Bacula devrait le trouver s'il est dans une librairie standard. Sinon, et si l'option --with-readline n'est pas renseignée, readline sera désactivé. Cette option affecte la compilation de Bacula. Readline fournit le programme Console avec un historique des lignes de commandes et des capacités d'édition. Readline n'est désormais plus supporté, ce qui signifie que vous l'utilisez à vos risques et périls

--enable-readline Active le support readline. Désactivé par défaut en raison de nombreux problèmes de configuration, et parce que le paquetage semble devenir incompatible.

--with-tcp-wrappers=<path> Cette option précise que vous voulez TCP wrappers (man hosts_access(5)) compilé dans Bacula. Le chemin est facultatif puisque Bacula devrait, en principe, trouver les librairies dans les répertoires standards. Cette option affecte la compilation. Lorsque vous spécifierez vos restrictions dans les fichiers **/etc/hosts.allow** ou **/etc/hosts.deny**, n'utilisez pas l'option **twist** (man hosts_options(5)) ou le processus Bacula sera stoppé.

Pour plus d'informations sur la configuration et les tests de TCP wrappers, consultez la section Configurer et Tester TCP Wrappers du chapitre sur la sécurité.

--with-working-dir=<working-directory-path> Cette option est obligatoire et spécifie un répertoire dans lequel Bacula peut placer en toute sécurité les fichiers qui resteront d'une exécution à l'autre. Par exemple, si la base de données interne est utilisée, Bacula stockera ces fichiers dans ce répertoire. Cette option n'est utilisée que pour modifier les fichiers de configuration de Bacula. Vous pourrez éventuellement effectuer cette modification directement en les éditant plus tard. Le répertoire spécifié ici n'est pas automatiquement créé par le processus d'installation, aussi vous devez veiller à ce qu'il existe avant votre première utilisation de Bacula.

--with-base-port=<port=number> Bacula a besoin de trois ports TCP/IP pour fonctionner (un pour la Console, un pour le Storage Daemon et un pour le File Daemon). L'option **--with-baseport** permet d'assigner automatiquement trois ports consécutifs à partir du port de base spécifié. Vous pouvez aussi changer les numéros de ports dans les fichiers de configuration. Cependant, vous devez prendre garde à ce que les numéros de ports se correspondent fidèlement dans chacun des trois fichiers de configuration. Le port de base par défaut est 9101, ce qui assigne les ports 9101 à 9103. Ces ports (9101, 9102 et 9103) ont été officiellement assigné à Bacula par l'IANA. Cette option n'est utilisée que pour modifier les fichiers de configuration de Bacula. Vous pouvez à tout moment faire cette modification en éditant directement ces fichiers.

--with-dump-email=<email-address> Cette option spécifie l'adresse e-mail qui recevra tous les *core dump*. Cette option n'est en principe utilisée que par les développeurs.

--with-pid-dir=<PATH> Ceci précise le répertoire de stockage du fichier d'id de processus lors de l'exécution. La valeur par défaut est : **/var/run**. Le répertoire spécifié ici n'est pas automatiquement créé par le processus d'installation, aussi vous devez veiller à ce qu'il existe avant votre première utilisation de Bacula.

--with-subsys-dir=<PATH> Cette option précise le répertoire de stockage des fichiers verrous du sous-système lors de l'exécution. Le répertoire par défaut est **/var/run/subsys**. Veillez à ne pas spécifier le même répertoire que pour l'option **sbindir**. Ce répertoire n'est utilisé que par les scripts de démarrage automatique. Le répertoire spécifié ici n'est pas automatiquement créé par le processus d'installation, aussi vous devez veiller à ce qu'il existe avant votre première utilisation de Bacula.

--with-dir-password=<Password> Cette option vous permet de préciser le mot de passe d'accès au Director (contacté, en principe,

depuis la console). S'il n'est pas précisé, configure en créé un aléatoirement.

--with-fd-password=<Password> Cette option vous permet de préciser le mot de passe d'accès au File Daemon (contacté, en principe, depuis le Director). S'il n'est pas précisé, configure en créé un aléatoirement.

--with-sd-password=<Password> Cette option vous permet de préciser le mot de passe d'accès au Storage Daemon (contacté, en principe, depuis le File Daemon). S'il n'est pas précisé, configure en créé un aléatoirement.

--with-dir-user=<User> Cette option vous permet de spécifier l'UserId utilisé pour l'exécution du Director. Le Director doit être démarré en tant que root, mais n'a pas besoin d'être exécuté en tant que root. Après avoir effectué les opérations d'initialisation préliminaires, il peut redescendre au niveau de l'UserId spécifié dans cette option. Si vous utilisez cette option, vous devez créer l'utilisateur User avant d'exécuter **make install**, car le répertoire de travail de Bacula appartiendra à cet utilisateur.

--with-dir-group=<Group> Cette option vous permet de spécifier le GroupId utilisé pour l'exécution du Director. Le Director doit être démarré en tant que root, mais n'a pas besoin d'être exécuté en tant que root. Après avoir effectué les opérations d'initialisation préliminaires, il peut redescendre au niveau du GroupId spécifié dans cette option. Si vous utilisez cette option, vous devez créer le groupe Group avant d'exécuter **make install**, car le répertoire de travail de Bacula appartiendra à ce groupe.

--with-sd-user=<User> Cette option vous permet de spécifier l'UserId utilisé pour exécuter le Storage Daemon. Le Storage Daemon doit être démarré en tant que root, mais n'a pas besoin d'être exécuté en tant que root. Après avoir effectué les opérations d'initialisation préliminaires, il peut redescendre au niveau de l'UserId spécifié dans cette option. Si vous utilisez cette option, veillez à ce que le Storage Daemon ait accès à tous les périphériques de stockage dont il a besoin.

--with-sd-group=<Group> Cette option vous permet de spécifier le GroupId utilisé pour exécuter le Storage Daemon. Le Storage Daemon doit être démarré en tant que root, mais n'a pas besoin d'être exécuté en tant que root. Après avoir effectué les opérations d'initialisation préliminaires, il peut redescendre au niveau du GroupId spécifié dans cette option.

--with-fd-user=<User> Cette option vous permet de spécifier l'UserId utilisé pour exécuter le File Daemon. Le File Daemon

doit être démarré et, dans la plupart des cas, exécuté en tant que root, de sorte que cette option n'est utilisée que dans des cas bien particuliers. Malgré tout, après avoir effectué les opérations d'initialisation préliminaires, il peut redescendre au niveau de l'UserId spécifié dans cette option.

--with-fd-group=<Group> Cette option vous permet de spécifier le GroupId utilisé pour exécuter le File Daemon. Le File Daemon doit être démarré et, dans la plupart des cas, exécuté en tant que root, de sorte que cette option n'est utilisée que dans des cas bien particuliers. Malgré tout, après avoir effectué les opérations d'initialisation préliminaires, il peut redescendre au niveau du GroupId spécifié dans cette option.

Notez: de nombreuses options supplémentaires vous sont présentées lorsque vous entrez **./configure --help**, mais elles ne sont pas implémentées.

Options recommandées pour la plupart des systèmes

Pour la plupart des systèmes, nous recommandons de commencer avec les options suivantes :

```
./configure \
  --enable-smartalloc \
  --sbindir=$HOME/bacula/bin \
  --sysconfdir=$HOME/bacula/bin \
  --with-pid-dir=$HOME/bacula/bin/working \
  --with-subsys-dir=$HOME/bacula/bin/working \
  --with-mysql=$HOME/mysql \
  --with-working-dir=$HOME/bacula/working
```

Si vous souhaitez installer Bacula dans un répertoire d'installation plutôt que de l'exécuter depuis le répertoire de compilation, (comme le feront les développeurs la plupart du temps), vous devriez aussi inclure les options --sbindir et --sysconfdir avec les chemins appropriés. Aucune n'est nécessaire si vous ne vous servez pas de "make install", comme c'est le cas pour la plupart des travaux de développement. Le processus d'installation va créer les répertoires sbindir et sysconfdir s'ils n'existent pas, mais il ne créera pas les répertoires pid-dir, subsys-dir ni working-dir, aussi assurez vous qu'ils existent avant de lancer Bacula. L'exemple ci-dessous montre la façon de procéder de Kern.

RedHat

Avec SQLite:

```
CFLAGS="-g -Wall" ./configure \
  --sbindir=$HOME/bacula/bin \
  --sysconfdir=$HOME/bacula/bin \
  --enable-smartalloc \
  --with-sqlite=$HOME/bacula/depkgs/sqlite \
  --with-working-dir=$HOME/bacula/working \
  --with-pid-dir=$HOME/bacula/bin/working \
  --with-subsys-dir=$HOME/bacula/bin/working \
  --enable-gnome \
  --enable-conio
```

ou

```
CFLAGS="-g -Wall" ./configure \
  --sbindir=$HOME/bacula/bin \
  --sysconfdir=$HOME/bacula/bin \
  --enable-smartalloc \
  --with-mysql=$HOME/mysql \
  --with-working-dir=$HOME/bacula/working
  --with-pid-dir=$HOME/bacula/bin/working \
  --with-subsys-dir=$HOME/bacula/bin/working
  --enable-gnome \
  --enable-conio
```

ou une installation RedHat complètement traditionnelle :

```
CFLAGS="-g -Wall" ./configure \
  --prefix=/usr
  --sbindir=/usr/sbin \
  --sysconfdir=/etc/bacula \
  --with-scriptdir=/etc/bacula \
  --enable-smartalloc \
  --enable-gnome \
  --with-mysql\
  --with-working-dir=/var/bacula
  --with-pid-dir=$HOME/var/run \
  --with-subsys-dir=/var/lock/subsys
  --enable-conio
```

Notez que Bacula suppose que les répertoires /var/bacula, /var/run et /var/lock/subsys existent, ils ne seront pas crées par le processus d'installation.
D'autre part, avec gcc 4.0.1 20050727 (Red Hat 4.0.1-5) sur processeur AMD64 et sous CentOS4 64 bits, un bug du compilateur génère du code erroné qui conduit Bacula à des erreurs de segmentation. Typiquement, vous le rencontrerez d'abord avec le Storage Daemon. La solution consiste à s'assurer que Bacula est compilé sans optimisation (normalement -O2)

Solaris

Pour installer Bacula depuis les sources, il vous faudra les paquetages suivants sur votre système (ils ne sont pas installés par défaut) : libiconv, gcc 3.3.2, stdc++, libgcc (pour les librairies stdc++ and gcc_s), make 3.8 ou plus récent.
Il vous faudra probablement aussi ajouter /usr/local/bin et /usr/css/bin à PATH pour ar.

```
#!/bin/sh
CFLAGS="-g" ./configure \
  --sbindir=$HOME/bacula/bin \
  --sysconfdir=$HOME/bacula/bin \
  --with-mysql=$HOME/mysql \
  --enable-smartalloc \
  --with-pid-dir=$HOME/bacula/bin/working \
  --with-subsys-dir=$HOME/bacula/bin/working \
  --with-working-dir=$HOME/bacula/working
```

Comme mentionné ci-dessus, le processus d'installation va créer les répertoires sbindir et sysconfdir s'ils n'existent pas, mais il ne créera pas les répertoires pid-dir, subsys-dir ni working-dir, aussi assurez vous qu'ils existent avant de lancer Bacula.
Notez que vous pouvez aussi avoir besoin des paquetages suivants pour installer Bacula depuis les sources :

```
SUNWbinutils,
SUNWarc,
SUNWhea,
SUNWGcc,
SUNWGnutls
SUNWGnutls-devel
SUNWGmake
SUNWgccruntime
SUNWlibgcrypt
SUNWzlib
SUNWzlibs
SUNWbinutilsS
SUNWGmakeS
SUNWlibm

export
PATH=/usr/bin::/usr/ccs/bin:/etc:/usr/openwin/bin:/usr/local/bin:/usr/sfw/bin:/opt/sfw/bin:/usr/ucb
```

FreeBSD

Veuillez consulter: The FreeBSD Diary pour une description détaillée de la méthode pour faire fonctionner Bacula sur votre système. De plus, les utilisateurs de versions de FreeBSD antérieures à la 4.9-STABLE du lundi 29 décembre 2003 15:18:01 qui envisagent d'utiliser des lecteurs de bandes doivent consulter le chapitre Tester son lecteur de bandes

de ce manuel pour d'**importantes** informations sur la configuration des lecteurs pour qu'ils soient compatibles avec Bacula.
Si vous utilisez Bacula avec MySQL, vous devriez prendre garde à compiler MySQL avec les threads natifs de FreeBSD plutôt qu'avec ceux de Linux, car c'est avec ceux là qu'est compilé Bacula et le mélange des deux ne fonctionnera probablement pas.

Win32

Pour installer la version binaire Win32 du File Daemon, consultez le chapitre Installation sur systèmes Win32 de ce document.

Systèmes Windows avec CYGWIN installé

A partir de la version 1.34, Bacula n'utilise plus CYGWIN pour le client Win32. Il est cependant encore compilé sous un environnement CYGWIN – Bien que vous puissiez probablement le faire avec seulement VC Studio. Si vous souhaitez compiler le client Win32 depuis les sources, il vous faudra Microsoft C++ version 6.0 ou supérieur. Dans les versions de Bacula antérieures à la 1.33, CYGWIN était utilisé.
Notez qu'en dépit du fait que la plupart des éléments de Bacula puissent compiler sur les systèmes Windows, la seule partie que nous avons testée et utilisée est le File Daemon.
Finalement, vous devriez suivre les instructions d'installation de la section Win32 Installation sur systèmes Win32 de ce document en occultant la partie qui décrit la décompression de la version binaire.

Le script Configure de Kern

Voici le script que j'utilise pour compiler sur mes machines Linux de "production":

```
#!/bin/sh
# This is Kern's configure script for Bacula
CFLAGS="-g -Wall" \
  ./configure \
    --sbindir=$HOME/bacula/bin \
    --sysconfdir=$HOME/bacula/bin \
    --enable-smartalloc \
    --enable-gnome \
    --with-pid-dir=$HOME/bacula/bin/working \
    --with-subsys-dir=$HOME/bacula/bin/working \
    --with-mysql=$HOME/mysql \
    --with-working-dir=$HOME/bacula/bin/working \
    --with-dump-email=$USER \
    --with-smtp-host=mail.your-site.com \
    --with-baseport=9101
exit 0
```

Notez que je fixe le port de base à 9101, ce qui signifie que Bacula utilisera le port 9101 pour la console Director, le port 9102 pour le File Daemon, et le 9103 pour le Storage Daemon. Ces ports devraient être disponibles sur tous les systèmes étant donné qu'ils ont été officiellement attribués à Bacula par l'IANA (Internet Assigned Numbers Authority). Nous recommandons fortement de n'utiliser que ces ports pour éviter tout conflit avec d'autres programmes. Ceci est en fait la configuration par défaut si vous n'utilisez pas l'option **--with-baseport**.
Vous pouvez aussi insérer les entrées suivantes dans votre fichier **/etc/services** de façon à rendre les connections de Bacula plus aisées à repérer (i.e. netstat -a):

```
bacula-dir      9101/tcp
bacula-fd       9102/tcp
bacula-sd       9103/tcp
```

Installer Bacula

Avant de personnaliser vos fichiers de configuration, vous voudrez installer Bacula dans son répertoire définitif. tapez simplement:

```
make install
```

Si vous avez précédemment installé Bacula, les anciens binaires seront écrasés, mais les anciens fichiers de configuration resteront inchangés, et les "nouveaux" recevront l'extension **.new**. Généralement, si vous avez déjà installé et exécuté Bacula, vous préfèrerez supprimer ou ignorer les fichiers de configuration avec l'extension **.new**

Compiler un File Daemon (ou Client)

Si vous exécutez le Director et le Storage Daemon sur une machine et si vous voulez sauvegarder une autre machine, vous devez avoir un File Daemon sur cette machine. Si la machine et le système sont identiques, vous pouvez simplement copier le binaire du File Daemon **bacula-fd** ainsi que son fichier de configuration **bacula-fd.conf**, puis modifier le nom et le mot de passe dans **bacula-fd.conf** de façon à rendre ce fichier unique. Veillez à faire les modifications correspondantes dans le fichier de configuration du Director (**bacula-dir.conf**).
Si les architectures, les systèmes, ou les versions de systèmes diffèrent, il vous faudra compiler un File Daemon sur la machine cliente. Pour ce faire, vous pouvez utiliser la même commande **./configure** que celle utilisée pour construire le programme principal, soit en partant d'une copie fraiche du répertoire des sources, soit en utilisant **make distclean** avant de lancer **./configure**.

Le File Daemon n'ayant pas d'accès au catalogue, vous pouvez supprimer les option **--with-mysql** ou **--with-sqlite**. Ajoutez l'option **--enable-client-only**. Ceci va compiler seulement les librairies et programmes clients, et donc éviter d'avoir à installer telle ou telle base de données. Lancez make avec cette configuration, et seul le client sera compilé.

Démarrage automatique des Daemons

Si vous souhaitez que vos *daemons* soient lancés automatiquement au démarrage de votre système (une bonne idée !), une étape supplémentaire est requise. D'abord, le processus ./configure doit reconnaître votre système – ce qui signifie que ce doit être une plate-forme supportée et non **inconnue**, puis vous devez installer les fichiers dépendants de la plate-forme comme suit :

```
(devenez root)
make install-autostart
```

Notez que la fonction d'autodémarrage n'est implémentée que pour les systèmes que nous supportons officiellement (actuellement FreeBSD, RedHat Linux, et Solaris), et n'a été pleinement testée que sur RedHat Linux.
make install-autostart installe les scripts de démarrage apropriés ainsi que les liens symboliques nécessaires. Sur RedHat Linux, Ces scripts résident dans **/etc/rc.d/init.d/bacula-dir /etc/rc.d/init.d/bacula-fd**, et **/etc/rc.d/init.d/bacula-sd**. Toutefois, leur localisation exacte dépend de votre système d'exploitation. Si vous n'installez que le File Daemon, tapez:

```
make install-autostart-fd
```

Autres notes concernant la compilation

Pour recompiler tout exécutable, tapez

```
make
```

dans le répertoire correspondant.. Afin d'éliminer tous les objets et binaires (y compris les fichiers temporaires nommés 1,2 ou 3 qu'utilise Kern), tapez

```
make clean
```

Pour un nettoyage exhaustif en vue de distribution, entrez:

```
make distclean
```

Notez que cette commande supprime les Makefiles. Elle est en principe lancée depuis la racine du répertoire des sources pour les préparer à la distribution. Pour revenir de cet état, vous devez réexécuter la commande **./configure** à la racine des sources puisque tous les Makefiles ont été détruits.
Pour ajouter un nouveau fichier dans un sous-répertoire, éditez Makefile.in dans ce sous-répertoire, puis faites un simple **make**. Dans la plupart des cas, le make reconstruira le Makefile à partir du nouveau Makefile.in. Dans certains cas, il peut être nécessaire d'exécuter **make** une deuxième fois. Dans les cas extrèmes, remontez à la racine des sources et entrez **make Makefiles**.
Pour ajouter des dépendances:

```
make depend
```

La commande **make depend** insère les fichiers d'en-têtes de dépendances aux Makefile et Makefile.in pour chaque fichier objet. Cette commande devrait être lancée dans chaque répertoire où vous modifiez les dépendances. En principe, il suffit de l'exécuter lorsque vous ajoutez ou supprimez des sources ou fichiers d'en-têtes. **make depend** est invoqué automatiquement durant le processus de configuration.
Pour installer:

```
make install
```

En principe, vous n'utilisez pas cette commande si vous êtes en train de développer Bacula, mais si vous vous apprétez à l'exécuter pour sauvegarder vos systèmes.
Après avoir lancé **make install**, les fichiers suivants seront installés sur votre système (à peu de choses près). La liste exacte des fichiers installés et leur localisation dépend de votre commande **c./configure** (e.g. gnome-console et gnome-console.conf ne sont pas installés si vous ne configurez pas GNOME. De même, si vous utilisez SQLite plutôt que MySQL, certains fichiers seront différents.

```
bacula
bacula-dir
bacula-dir.conf
bacula-fd
bacula-fd.conf
bacula-sd
bacula-sd.conf
bacula-tray-monitor
tray-monitor.conf
bextract
bls
bscan
```

```
btape
btraceback
btraceback.gdb
bconsole
bconsole.conf
create_mysql_database
dbcheck
delete_catalog_backup
drop_bacula_tables
drop_mysql_tables
fd
gnome-console
gnome-console.conf
make_bacula_tables
make_catalog_backup
make_mysql_tables
mtx-changer
query.sql
bsmtp
startmysql
stopmysql
wx-console
wx-console.conf
```

Installer Tray Monitor

Le Tray Monitor est déjà installé si vous avez utilisé l'option **--enable-tray-monitor** de la commande configure et exécuté **make install**.
Comme vous n'exécutez pas votre environnement graphique en tant que root (si vous le faites, vous devriez changer cette mauvaise habitude), n'oubliez pas d'autoriser votre utilisateur à lire **tray-monitor.conf**, et exécuter **bacula-tray-monitor** (ceci ne constitue pas une faille de sécurité).
Puis, connectez vous à votre environnement graphique (KDE, Gnome, ou autre), lancez **bacula-tray-monitor** avec votre utilisateur et observez si l'icone d'une cartouche apparaît quelque part sur l'écran, usuellement dans la barre des tâches. Sinon, suivez les instructions suivantes relatives à votre gestionnaire de fenêtres.

GNOME

System tray, ou zone de notification si vous utilisez la terminologie GNOME, est supporté par GNOME depuis la version 2.2. Pour l'activer, faites un click droit sur un de vos espaces de travail, ouvrez le menu **Ajouter à ce bureau**, puis **Utilitaire** et enfin, cliquez sur **Zone de notification**. (NDT: A valider)

KDE

System tray est supporté par KDE depuis la version 3.1. Pour l'activer, faites un click droit sur la barre de tâches, ouvrez le menu **Ajouter**, puis **Applet**, enfin cliquez sur **System Tray**.

Autres gestionnaires de fenêtres

Lisez la documentation pour savoir si votre gestionnaire de fenêtres supporte le standard *systemtray* de FreeDesktop, et comment l'activer le cas échéant.

Modifier les fichiers de configuration de Bacula

Consultez le chapitre Configurer Bacula de ce manuel pour les instructions de configuration de Bacula.

Une brève documentation

Ce chapitre vous guidera à travers les étapes nécessaires pour exécuter Bacula. Pour cela, nous supposons que vous avez installé Bacula, peut être dans un simple répertoire comme le décrit le chapitre précédent, auquel cas vous pouvez exécuter Bacula sans être root pour ces tests. Nous supposons d'autre part que vous n'avez pas modifié les fichiers de configuration. Dans le cas contraire, nous vous recommandons de désinstaller Bacula et de le réinstaller sans rien modifier. Les exemples de ce chapitre utilisent les fichiers de configuration par défaut, et créent les volumes dans le répertoire **/tmp** de votre disque. De plus, les données sauvegardées seront celle du répertoire des sources de Bacula où vous l'avez compilé. Par conséquent, tous les *daemons* peuvent être exécutés sans les droits root pour ces tests. Notez bien qu'en production, vos File Daemons devront être exécutés en tant que root. Voyez le chapitre sur la sécurité pour plus d'informations sur ce sujet.
Voici les étapes que nous suivrons :

1. cd <install-directory>
2. Démarrer la base de données (si vous utilisez MySQL ou PostgreSQL)
3. Démarrer les *daemons* avec **./bacula start**
4. Lancer le programme Console pour interagir avec le Director
5. Lancer un job
6. Lorsqu'un volume est plein, le démonter, s'il s'agit d'une cartouche, en étiqueter une nouvelle et poursuivre. Dans ce chapitre, nous n'écrirons que sur des volumes fichier, aussi vous n'avez pas à vous inquiéter au sujet des cartouches pour le moment.
7. Tester la restauration de quelques fichiers depuis le volume fraichement écrit pour s'assurer de la validité de la sauvegarde et qu'il est possible de restaurer. Mieux vaut essayer avant qu'un désastre ne survienne...
8. Ajouter un second client.

Chacune de ces étapes est décrite en détail ci-dessous.

Avant d'exécuter Bacula

Avant d'utiliser Bacula pour la première fois en production, nous vous recommandons d'exécuter la commande du programme ainsi qu'il est décrit dans le chapitre Programmes utilitaires de ce manuel. Ce programme vous aidera à vous assurer que votre lecteur de bandes fonctionne correctement avec Bacula. Si vous avez un lecteur moderne de marque HP, Sony, ou Quantum DDS ou DLT qui fonctionne sous

Linux ou Solaris, vous pouvez probablement vous dispenser de faire ce test car Bacula est bien testé avec ces lecteurs et ces systèmes. Dans tous les autres cas, vous êtes **fortement** encouragé à exécuter les tests avant de poursuivre. **btape** dispose aussi d'une commande **fill** qui tente de reproduire le comportement de Bacula lorsqu'il remplit une cartouche et qu'il poursuit son écriture sur la suivante. Vous devriez songer à faire ce test, sachez cependant qu'il peut être long (environ 4 heures sur mon lecteur) de remplir une cartouche de haute capacité.

Démarrer la base de données

Si vous utilisez MySQL ou PostgreSQL pour votre catalogue Bacula, vous devez démarrer la base de données avant d'essayer de lancer un job pour éviter d'obtenir des messages d'erreur au démarrage de Bacula. J'utilise les scripts **startmysql** et **stopmysql** pour démarrer mon MySQL local. Notez que si vous utilisez SQLite, vous n'aurez pas à utiliser **startmysql** ou **stopmysql**. Si vous utilisez ceci en production, vous souhaiterez probablement trouver un moyen pour démarrer automatiquement MySQL ou PostgreSQL après chaque redémarrage du système.
Si vous utilisez SQLite (c'est à dire, si vous avez spécifié l'option **--with-sqlite=xxx** de la commande **./configure**, vous n'avez rien à faire. SQLite est démarrée automatiquement par **Bacula**.

Démarrer les daemons

Que vous ayez compilé Bacula depuis les sources ou que vous ayez installé les rpms, tapez simplement :
./bacula start
dans votre répertoire d'installation pour démarrer les trois *daemons*.
Le script **bacula** lance le Storage Daemon, le File Daemon et le Director Daemon, qui tournent tous trois en tant que *daemons* en tâche de fond. Si vous utilisez la fonction de démarrage automatique de Bacula, vous pouvez, au choix, lancer les trois *daemons* lors du démarrage, ou au contraire les lancer individuellement avec les scripts **bacula-dir**, **bacula-fd**, et **bacula-sd** usuellement situés dans **/etc/init.d**, bien que leur localisation effective soit dépendante du système d'exploitation.
Notez que seul le File Daemon a été porté sur les systèmes Windows, et qu'il doit être démarré différamment. Veuillez consulter le chapitre La version Windows de Bacula de ce manuel.
Les paquetages rpm configurent les *daemons* pour qu'ils s'exécutent en tant qu'utilisateur root et en tant que groupe bacula. Le processus d'installation rpm se charge de créer le groupe bacula s'il n'existe

pas sur le système. Tout utilisateur ajouté au groupe bacula hérite de l'accès aux fichiers créés par les *daemons*. Pour modifier ce comportement, éditez les scripts de démarrage des *daemons* :

- /etc/bacula/bacula
- /etc/init.d/bacula-dir
- /etc/init.d/bacula-sd
- /etc/init.d/bacula-fd

puis redémarrez-les.
Le chapitre installation de ce manuel indique comment installer les scripts de démarrage automatique des *daemons*.

Interagir avec le Director pour l'interroger sur l'état de Bacula ou lancer des jobs

Pour communiquer avec le Director et pour s'enquérir de l'état de Bacula ou de jobs en cours d'exécution, tapez simplement :
./bconsole
dans le répertoire de plus haut niveau.
Si vous avez installé la console GNOME et utilisé l'option **--enable-gnome** de la commande configure, vous pouvez aussi utiliser la console GNOME en tapant :
./gnome-console
Vous pouvez aussi utiliser le programme wxWidgets **wx-console**.
Pour simplifier, nous ne décrirons ici que le programme **./bconsole**. La plus grande partie de ce qui est décrit ici s'applique aussi aux programmes **./gnome-console** et **wx-console**.
La commande **./bconsole** lance le programme Console, qui se connecte au Director. Bacula étant un programme réseau, vous pouvez utiliser la Console depuis n'importe quelle machine de votre réseau. Cependant, la plupart du temps le Console est exécutée sur la même machine que le Director. En principe, la Console devrait produire un affichage similaire à :

```
[kern@polymatou bin]$ ./bconsole
Connecting to Director lpmatou:9101
1000 OK: HeadMan Version: 1.30 (28 April 2003)
*
```

L'astérisque est l'invite de commande de la console.
Tapez **help** pour obtenir la liste des commandes disponibles :

```
*help
  Command    Description
  =======    ===========
```

```
  add        add media to a pool
  autodisplay autodisplay [on/off] -- console messages
  automount  automount [on/off] -- after label
  cancel     cancel job=nnn -- cancel a job
  create     create DB Pool from resource
  delete     delete [pool=<pool-name> | media volume=<volume-name>]
  estimate   performs FileSet estimate debug=1 give full listing
  exit       exit = quit
  help       print this command
  label      label a tape
  list       list [pools | jobs | jobtotals | media <pool> |
             files jobid=<nn>]; from catalog
  llist      full or long list like list command
  messages   messages
  mount      mount <storage-name>
  prune      prune expired records from catalog
  purge      purge records from catalog
  query      query catalog
  quit       quit
  relabel    relabel a tape
  release    release <storage-name>
  restore    restore files
  run        run <job-name>
  setdebug   sets debug level
  show       show (resource records) [jobs | pools | ... | all]
  sqlquery   use SQL to query catalog
  status     status [storage | client]=<name>
  time       print current time
  unmount    unmount <storage-name>
  update     update Volume or Pool
  use        use catalog xxx
  var        does variable expansion
  version    print Director version
  wait       wait until no jobs are running
*
```

Pour plus de détails sur les commandes de la console, consultez le chapitre Console de ce manuel.

exécuter un job

A ce stade, nous supposons que vous avez :

- Configuré Bacula avec la commande **./configure --your-options**
- Compilé Bacula avec la commande **make**
- Installé Bacula avec la commande **make install**
- Créé votre catalogue avec, par exemple, la commande **./create_sqlite_database**
- Créé les tables du catalogue avec la commande **./make_bacula_tables**

- Eventuellement édité votre fichier **bacula-dir.conf** pour le personnaliser quelque peu. ATTENTION ! Si vous modifiez le nom du Director ou son mot de passe, vous devez faire les modifications correspondantes dans les autres fichiers de configuration. Il est sans dout préférable, pour l'instant, de ne rien changer.
- Démarré Bacula avec la commande **./bacula start**
- Invoqué le programme Console avec la commande **./bconsole**.

En outre, nous supposons pour le moment que vous utilisez les fichiers de configuration par défaut.
Maintenant, entrez les commandes suivantes :

```
show filesets
```

Vous devriez obtenir quelque chose comme :

```
FileSet: name=Full Set
      O M
      N
      I /home/kern/bacula/regress/build
      N
      E /proc
      E /tmp
      E /.journal
      E /.fsck
      N
FileSet: name=Catalog
      O M
      N
      I /home/kern/bacula/regress/working/bacula.sql
      N
```

Il s'agit d'un **FileSet** prédéfini qui sauvegardera le répertoire des sources de Bacula. Les noms de répertoires qui seront réellement affichés devraient correspondre à votre configuration. Dans une perspective de tests, nous avons choisi un répertoire de taille et de complexité modérée (environ 40 Mo). Le FileSet **Catalog** est utilisé pour sauvegarder le catalogue Bacula et nous ne nous y attarderons pas pour le moment. Les entrées **I** sont les fichiers ou répertoires qui seront inclus dans la sauvegarde, tandis que les entrées **E** sont ceux qui en seront exclus, quand aux entrées **O**, ce sont les options spécifiées pour ce FileSet. Vous pouvez changer ce qui est sauvegardé en modifiant la ligne **File =** de la ressource **FileSet**.
Il est maintenant temps de lancer votre première sauvegarde. Nous allons sauvegarder votre répertoire sources de Bacula vers un volume File dans votre répertoire **/tmp** afin de vous montrer combien c'est facile. Saisissez :

```
status dir
```

Vous devriez obtenir :

```
rufus-dir Version: 1.30 (28 April 2003)
Daemon started 28-Apr-2003 14:03, 0 Jobs run.
Console connected at 28-Apr-2003 14:03
No jobs are running.
Level          Type     Scheduled          Name
===============================================================
Incremental    Backup   29-Apr-2003 01:05  Client1
Full           Backup   29-Apr-2003 01:10  BackupCatalog
====
```

Où les dates et le nom du Director seront différents et en accord avec votre installation. Ceci montre qu'une sauvegarde incrémentale est planifiée pour le job **Client1** à 1h05, et qu'une sauvegarde full est planifiée pour le job **BackupCatalog** à 1h10. Vous devriez remplacer le nom **Client1** par celui de votre machine, sinon vous risquez la confusion lorsque vous ajouterez de nouveaux clients. Pour ma machine réelle, j'utilise **Rufus** plutôt que **Client1**.
A présent, tapez :

```
status client
```

Vous devriez obtenir :

```
The defined Client resources are:
     1: rufus-fd
Item 1 selected automatically.
Connecting to Client rufus-fd at rufus:8102
rufus-fd Version: 1.30 (28 April 2003)
Daemon started 28-Apr-2003 14:03, 0 Jobs run.
Director connected at: 28-Apr-2003 14:14
No jobs running.
====
```

Dans ce cas, le client se nomme **rufus-fd**, votre nom sera différent, mais la ligne qui débute par **rufus-fd Version...** est produite par votre File Daemon, nous sommes donc maintenant surs qu'il fonctionne.
Finalement, faites de même pour votre Storage Daemon :

```
status storage
```

Vous devriez obtenir :

```
The defined Storage resources are:
     1: File
Item 1 selected automatically.
Connecting to Storage daemon File at rufus:8103
rufus-sd Version: 1.30 (28 April 2003)
Daemon started 28-Apr-2003 14:03, 0 Jobs run.
Device /tmp is not open.
No jobs running.
====
```

Vous noterez que le périphérique du Storage Daemon par défaut est nommé **File** et qu'il utilise le périphérique **/tmp**, qui n'est actuellement pas ouvert.
Maintenant, lancez un job :

```
run
```

Vous devriez obtenir :

```
Using default Catalog name=MyCatalog DB=bacula
A job name must be specified.
The defined Job resources are:
     1: Client1
     2: BackupCatalog
     3: RestoreFiles
Select Job resource (1-3):
```

Ici, Bacula affiche la liste des trois différents jobs que vous pouvez exécuter. Choisissez le numéro **1** et validez (entrée).
Vous devriez obtenir :

```
Run Backup job
JobName:  Client1
FileSet:  Full Set
Level:    Incremental
Client:   rufus-fd
Storage:  File
Pool:     Default
When:     2003-04-28 14:18:57
OK to run? (yes/mod/no):
```

Prenez un peu de temps pour examiner cet affichage et le comprendre. Il vous est demandé de valider, modifier ou annuler l'exécution d'un job nommé **Client1** avec le FileSet **Full Set** que nous avons affiché plus haut en incrémental sur votre client rufus, utilisant le périphérique de stockage **File** et le pool **Default** à la date indiquée sur la ligne "When".
Nous avons le choix de valider (**yes**), modifier un ou plusieurs des paramètres ci-dessus (**mod**), ou de ne pas exécuter le job (**no**).
Validez l'exécution du job (**yes**), vous devriez immédiatement obtenir l'invite de commande de la console (un astérisque). Après quelques minutes, la commande **messages** devrait produire un résultat tel que :

```
28-Apr-2003 14:22 rufus-dir: Last FULL backup time not found. Doing
                  FULL backup.
28-Apr-2003 14:22 rufus-dir: Start Backup JobId 1,
                  Job=Client1.2003-04-28_14.22.33
28-Apr-2003 14:22 rufus-sd: Job Client1.2003-04-28_14.22.33 waiting.
                  Cannot find any appendable volumes.
Please use the "label"  command to create a new Volume for:
    Storage:      FileStorage
    Media type:   File
    Pool:         Default
```

Le premier message signale qu'aucune sauvegarde full n'a jamais été faite, et que par conséquent Bacula élève votre incrémentale en une Full (ce comportement est normal). Le second message indique que le job a démarré avec le JobId 1 et le troisième message vous informe que Bacula ne peut trouver aucun volume dans le pool Default sur lequel écrire les données du job. Ceci est normal, car nous n'avons encore créé (ou étiqueté) aucun volume. Bacula vous fournit tous les détails concernant le volume dont il a besoin.
A ce point, le job est bloqué en attente d'un volume. Vous pouvez le vérifier en utilisant la commande **status dir**. Pour continuer, vous devez créer un volume sur lequel Bacula pourra écrire. Voici la manipulation :

```
label
```

Bacula devrait afficher :

```
The defined Storage resources are:
     1: File
Item 1 selected automatically.
Enter new Volume name:
```

Entrez un nom commenant par une lettre et ne contenant que des chiffres et des lettres (périodes, tirets et souligné "" *sontaussiautorisés*).*Parexempleentrez***TestVolume001**,*vousdevriezobtenir* :

```
Defined Pools:
     1: Default
Item 1 selected automatically.
Connecting to Storage daemon File at rufus:8103 ...
Sending label command for Volume "TestVolume001" Slot 0 ...
3000 OK label. Volume=TestVolume001 Device=/tmp
Catalog record for Volume "TestVolume002", Slot 0  successfully created.
Requesting mount FileStorage ...
3001 OK mount. Device=/tmp
```

Finalement, tapez la commande **messages**, vous devriez obtenir quelque chose comme :

```
28-Apr-2003 14:30 rufus-sd: Wrote label to prelabeled Volume
   "TestVolume001" on device /tmp
28-Apr-2003 14:30 rufus-dir: Bacula 1.30 (28Apr03): 28-Apr-2003 14:30
JobId:                  1
Job:                    Client1.2003-04-28_14.22.33
FileSet:                Full Set
Backup Level:           Full
Client:                 rufus-fd
Start time:             28-Apr-2003 14:22
End time:               28-Apr-2003 14:30
Files Written:          1,444
```

```
Bytes Written:          38,988,877
Rate:                   81.2 KB/s
Software Compression:   None
Volume names(s):        TestVolume001
Volume Session Id:      1
Volume Session Time:    1051531381
Last Volume Bytes:      39,072,359
FD termination status:  OK
SD termination status:  OK
Termination:            Backup OK
28-Apr-2003 14:30 rufus-dir: Begin pruning Jobs.
28-Apr-2003 14:30 rufus-dir: No Jobs found to prune.
28-Apr-2003 14:30 rufus-dir: Begin pruning Files.
28-Apr-2003 14:30 rufus-dir: No Files found to prune.
28-Apr-2003 14:30 rufus-dir: End auto prune.
```

Si rien ne se passe dans l'immédiat, vous pouvez continuer de rentrer la commande **messages** jusqu'à ce que le job se termine, ou utiliser la commande **autodisplay on** afin que les messages soient affichés dès-qu'ils sont disponibles.
si vous faites **ls -l** dans votre répertoire **/tmp**, vous verrez l'élément suivant :

```
-rw-r-----    1 kern     kern     39072153 Apr 28 14:30 TestVolume001
```

Il s'agit du volume File que vous venez juste d'écrire, et qui contient toutes les données du job que vous venez d'exécuter. Si vous exécutez d'autres jobs, il seront ajoutés à la suite de ce volume, à moins que vous n'ayez spécifié un autre comportement.
Vous vous demandez peut-être s'il va vous falloir étiqueter vous même chaque volume que Bacula sera amené à utiliser. La réponse, en ce qui concerne les volumes disque tels que celui que nous avons utilisé, est non. Il est possible de paramétrer Bacula pour qu'il créée lui même les volumes. En revanche, pour les volumes de type cartouche, il vous faudra très probablement étiqueter chaque volume que vous voulez utiliser.
Si vous souhaitez en rester là, saisissez simplement **quit** dans la console, puis stoppez Bacula avec **./bacula stop**. Pour nettoyer votre installation des résultats de vos tests, supprimez le fichier **/tmp/TestVolume001**, et réinitialiser votre catalogue en utilisant :

```
./drop_bacula_tables
./make_bacula_tables
```

Notez bien que ceci supprimera toutes les informations concernant les jobs précédemment exécutés et que, si c'est sans doute ce que vous souhaitez faire en fin de phase de test, ce n'est généralement pas une opération souhaitable en utilisation normale.
Si vous souhaitez essayer de restaurer les fichiers que vous venez de sauvegarder, lisez la section suivante.

Restaurer vos fichiers

Si vous avez utilisé la configuration par défaut et sauvegardé les sources de Bacula comme dans la démonstration ci-dessus, vous pouvez restaurer les fichiers sauvegardés en saisissant les commandes suivantes dans la Console :

```
restore all
```

Vous obtiendrez :

```
First you select one or more JobIds that contain files
to be restored. You will be presented several methods
of specifying the JobIds. Then you will be allowed to
select which files from those JobIds are to be restored.

To select the JobIds, you have the following choices:
     1: List last 20 Jobs run
     2: List Jobs where a given File is saved
     3: Enter list of comma separated JobIds to select
     4: Enter SQL list command
     5: Select the most recent backup for a client
     6: Select backup for a client before a specified time
     7: Enter a list of files to restore
     8: Enter a list of files to restore before a specified time
     9: Find the JobIds of the most recent backup for a client
    10: Find the JobIds for a backup for a client before a specified time
    11: Enter a list of directories to restore for found JobIds
    12: Cancel
Select item:  (1-12):
```

Comme vous pouvez le constater, les options sont nombreuses, mais pour l'instant, choisissez l'option **5** afin de sélectionner la dernière sauvegarde effectuée. Vous obtiendrez :

```
Defined Clients:
     1: rufus-fd
Item 1 selected automatically.
The defined FileSet resources are:
     1: 1  Full Set  2003-04-28 14:22:33
Item 1 selected automatically.
+-------+-------+----------+---------------------+---------------+
| JobId | Level | JobFiles | StartTime           | VolumeName    |
+-------+-------+----------+---------------------+---------------+
| 1     | F     | 1444     | 2003-04-28 14:22:33 | TestVolume002 |
+-------+-------+----------+---------------------+---------------+
You have selected the following JobId: 1
Building directory tree for JobId 1 ...
1 Job inserted into the tree and marked for extraction.
The defined Storage resources are:
     1: File
Item 1 selected automatically.
You are now entering file selection mode where you add and
remove files to be restored. All files are initially added.
Enter "done" to leave this mode.
```

```
cwd is: /
$
```

(J'ai tronqué l'affichage à droite par soucis de lisibilité.) Comme vous pouvez le constater au début de cet affichage, Bacula connaît vos clients, et puisque vous n'en avez qu'un, il est automatiquement sélectionné. Il en va de même pour le FileSet. Bacula produit alors une liste de tous les jobs qui constituent la sauvegarde courante. Dans le cas présent, il n'y en a qu'un. Notez que le Storage Daemon est aussi sélectionné automatiquement. Bacula est maintenant en mesure de produire une **arborescence** à partir de tous les fichiers qui ont été sauvegardés (il s'agit d'une représentation en mémoire de votre système de fichiers). A ce stade, vous pouvez utiliser les commandes **cd** , **ls** et **dir** pour naviguer dans l'arborescence et voir quels fichiers peuvent être restaurés. Par exemple, si je saisis **cd /home/kern/bacula/bacula-1.30** suivi de **dir**, j'obtiens la liste de tous les fichiers du répertoire source de Bacula. Pour plus d'information sur ce sujet, veuillez consulter le chapitre La commande Restore.
Pour quitter, tapez simplement :

```
done
```

Vous obtiendrez :

```
Bootstrap records written to
   /home/kern/bacula/testbin/working/restore.bsr
The restore job will require the following Volumes:

   TestVolume001
1444 files selected to restore.
Run Restore job
JobName:    RestoreFiles
Bootstrap:  /home/kern/bacula/testbin/working/restore.bsr
Where:      /tmp/bacula-restores
Replace:    always
FileSet:    Full Set
Client:     rufus-fd
Storage:    File
JobId:      *None*
When:       2005-04-28 14:53:54
OK to run? (yes/mod/no):
```

Si vous acceptez (**yes**), vos fichiers seront restaurés vers le répertoire **/tmp/bacula-restores**. Si vous préférez restaurer les fichiers à leurs emplacements d'origine, vous devez utiliser l'option **mod** et régler explicitement le paramètre **Where** à vide ou "/". Nous vous conseillons de poursuivre avec **yes**. Après quelques instants, la commande **messages** devrait produire la liste des fichiers restaurés, ainsi qu'un résumé du job qui devrait ressembler à ceci :

```
28-Apr-2005 14:56 rufus-dir: Bacula 1.30 (28Apr03): 28-Apr-2003 14:56
```

```
JobId:                  2
Job:                    RestoreFiles.2005-04-28_14.56.06
Client:                 rufus-fd
Start time:             28-Apr-2005 14:56
End time:               28-Apr-2005 14:56
Files Restored:         1,444
Bytes Restored:         38,816,381
Rate:                   9704.1 KB/s
FD termination status:  OK
Termination:            Restore OK
28-Apr-2005 14:56 rufus-dir: Begin pruning Jobs.
28-Apr-2005 14:56 rufus-dir: No Jobs found to prune.
28-Apr-2005 14:56 rufus-dir: Begin pruning Files.
28-Apr-2005 14:56 rufus-dir: No Files found to prune.
28-Apr-2005 14:56 rufus-dir: End auto prune.
```

Après avoir quitté la Console, vous pouvez examiner les fichiers dans le répertoire **/tmp/bacula-restores**, il contient l'arborescence avec tous vos fichiers. Supprimez-le après avoir vérifié :

```
rm -rf /tmp/bacula-restore
```

Quitter le programme Console

Saisissez simplement la commande **quit**.

Ajouter un client

Si vous êtes parvenus à faire fonctionner tous les exemples ci-dessus, vous êtes sans doute prèt à ajouter un nouveau client (File Daemon), c'est à dire une seconde machine que vous souhaitez sauvegarder. La seule chose à installer sur la nouvelle machine est le binaire **bacula-fd** (ou **bacula-fd.exe** pour Windows) et son fichier de configuration **bacula-fd.conf**. Vous pouvez démarrer en copiant le fichier précédemment créé moyennant une modification mineure pour l'adapter au nouveau client : changez le nom de File Daemon (**rufus-fd** dans l'exemple ci-dessus) en le nom que vous avez choisi pour le nouveau client. Le mieux est d'utiliser le nom de la machine. Par exemple :

```
...
#
# "Global" File daemon configuration specifications
#
FileDaemon {                          # this is me
  Name = rufus-fd
  FDport = 9102                  # where we listen for the director
  WorkingDirectory = /home/kern/bacula/working
  Pid Directory = /var/run
}
...
```

devient :

```
...
#
# "Global" File daemon configuration specifications
#
FileDaemon {                          # this is me
  Name = matou-fd
  FDport = 9102                  # where we listen for the director
  WorkingDirectory = /home/kern/bacula/working
  Pid Directory = /var/run
}
...
```

Où **rufus-fd** est devenu **matou-fd** (je ne montre qu'une partie du fichier). Le choix des noms vous appartient. Pour l'instant, je vous recommande de ne rien changer d'autre. Plus tard, vous changerez le mot de passe.
Installez cette configuration sur votre seconde machine. Il vous faut maintenant ajouter quelques lignes à votre **bacula-dir.conf** pour définir le nouveau File Daemon. En vous basant sur l'exemple initial qui devrait être installé sur votre système, ajoutez les lignes suivantes (essentiellement, une copie des lignes existantes avec seulement les noms modifiés) à votre **bacula-dir.conf** :

```
#
# Define the main nightly save backup job
#   By default, this job will back up to disk in /tmp
Job {
  Name = "Matou"
  Type = Backup
  Client = matou-fd
  FileSet = "Full Set"
  Schedule = "WeeklyCycle"
  Storage = File
  Messages = Standard
  Pool = Default
  Write Bootstrap = "/home/kern/bacula/working/matou.bsr"
}
# Client (File Services) to backup
Client {
  Name = matou-fd
  Address = matou
  FDPort = 9102
  Catalog = MyCatalog
  Password = "xxxxx"                  # password for
  File Retention = 30d                # 30 days
  Job Retention = 180d                # six months
  AutoPrune = yes                     # Prune expired Jobs/Files
}
```

Assurez-vous que le paramètre Address de la ressource Storage a pour valeur le nom pleinement qualifié et non quelque chose comme "localhost".

L'adresse spécifiée est envoyée au client et doit être un nom pleinement qualifié. Si vous utilisez "localhost", l'adresse du Storage Daemon ne sera pas résolue correctement, il en résultera un *timeout* lorsque le File Daemon échouera à connecter le Storage Daemon.
Il n'y a rien d'autre à faire. J'ai copié les ressources existantes pour créer un second job (Matou) pour sauvegarder le second client (matou-fd). le client se nomme **matou-fd** et le job **Matou**, le fichier bootstrap est modifié mais tout le reste est inchangé. Ceci signifie que Matou sera sauvegardé avec la même planification sur les mêmes cartouches. Vous pourrez changer ceci plus tard, pour le moment, restons simples.
La seconde modification consiste en l'ajout d'une nouvelle ressource Client qui définit **matou-fd** et qui a l'adresse correcte **matou** (mais dans la vraie vie, vous pouvez avoir besoin d'un nom pleinement qualifié ou d'une adresse IP. J'ai aussi conservé le même mot de passe (xxxxx dans l'exemple).
A ce stade, il suffit de redémarrer Bacula pour qu'il prenne en compte vos modifications. L'invite que vous avez vu plus haut devrait maintenant inclure la nouvelle machine.
Pour une utilisation en production vous voudrez probablement utiliser plusieurs pools et différentes planifications. Il vous appartient de faire les adaptations qui seyent à vos besoins. Dans tous les cas, n'oubliez pas de changer les mots de passe dans les fichiers de configuration du Director et du Client pour des raisons de sécurité.
Vous trouverez des astuces importantes concernant le changement des noms et mots de passe, ainsi qu'un diagramme décrivant leurs correspondances dans la section Erreurs d'authentification du chapitre FAQ de ce manuel.

Lorsque la cartouche est pleine

Si vous avez planifié votre job, il viendra un moment où la cartouche sera pleine et où **Bacula** ne pourra continuer. Dans ce cas, Bacula vous enverra un message tel que :

```
rufus-sd: block.c:337 === Write error errno=28: ERR=No space left
          on device
```

Ceci indique que Bacula a reu une erreur d'écriture à cause de la carouche pleine. Bacula va maintenant rechercher une cartouche utilisable dans le pool spécifié pour le job. Dans la situation idéale, vous avez réglé correctement vos rétentions et spécifié que vos cartouches peuvent être recyclées automatiquement. Dans ce cas, Bacula recycle automatiquement vos cartouches sorties de rétention et est en mesure de réécrire dessus. Pour plus d'informations sur le recyclage, veuillez consulter le chapitre Recyclage de ce manuel.
Si vous constatez que vos cartouches ne sont pas recyclées correctement, consultez la section sur le Recyclage manuel du chapitre Recyclage.

Si comme moi, vous avez un très grand nombre de cartouches que vous étiquetez avec la date de première écriture, si vous n'avez pas réglé vos périodes de rétention, Bacula ne trouvera pas de cartouche dans le pool et il vous enverra un message tel que :

```
rufus-sd: Job kernsave.2002-09-19.10:50:48 waiting. Cannot find any
          appendable volumes.
Please use the "label"  command to create a new Volume for:
    Storage:      SDT-10000
    Media type:   DDS-4
    Pool:         Default
```

Ce message sera répété une heure plus tard, puis deux heures plus tard et ainsi de suite en doublant à chaque fois l'intervalle à concurrence d'un jour jusqu'à ce que vous créiez un volume.
Que faire dans cette situation ?
La réponse est simple : d'abord, fermez le lecteur à l'aide de la commande **unmount** du programme Console. Si vous n'avez qu'un lecteur, il sera sélectionné automatiquement, sinon assurez-vous de démonter celui spécifié dans le message (dans ce cas **STD-10000**).
Ensuite, retirez la cartouche du lecteur et insérez-en une vierge. Notez que sur certains lecteurs anciens, il peut être nécessaire d'écrire une marque de fin de fichier (**mt -f /dev/nst0 weof**) pour éviter que le lecteur ne déroule toute la cartouche lorsque Bacula tente de lire le label. (NDT : j'ai un doute, la vo dit : "to prevent the drive from running away when Bacula attempts to read the label.")
Finalement, utilisez la commande **label** dans la console pour écrire un label sur le nouveau volume. la commande **label** va contacter le Storage Daemon pour qu'il écrive l'étiquette logicielle. Si cette opération se termine correctement, le nouveau volume est ajouté au pool et la commande **mount** est envoyée au Storage Daemon. Voyez les sections précédentes de ce chapitre pour plus de détails sur l'étiquetage des cartouches.
Bacula peut maintenant poursuivre le job et continuer d'écrire les données sauvegardées sur le nouveau volume.
Si Bacula cycle sur un pool de volumes, au lieu du message ci-dessus "Cannot find any appendable volumes.", Bacula peut vous demander de monter un volume particulier. Dans ce cas, essayez de le satisfaire. Si, pour quelque raison, vous n'avez plus le volume, vous pouvez monter n'importe quel autre volume du pool, pourvu qu'il soit utilisable, Bacula l'utilisera. La commande **list volumes** du programme Console permet de déterminer les volumes utilisables et ceux qui ne le sont pas.
Si, comme moi, vous avez paramétré correctement vos périodes de rétention, mais n'avez plus aucun volume libre, vous pouvez ré-étiqueter et ré-utiliser un volume comme suit :

- Saisissez **list volumes** dans la console et sélectionnez le volume

le plus anciens pour le ré-étiqueter.

- Si vos périodes de rétention sont judicieusement choisies, le volume devrait avoir le statut **Purged**.
- Si le statut n'est pas **Purged**, il vous faut purger le catalogue des jobs écrits sur ce volume. Ceci peut être fait avec la commande **purge jobs volume** dans la console. Si vous avez plusieurs pools, vous serez invité à choisir lequel avant de devoir saisir le VolumeName (ou MediaId).
- Enfin, utilisez simplement la commande **relabel** pour ré-étiqueter le volume.

Pour ré-étiqueter manuellement le volume, suivez les étapes supplémentaire ci-dessous :

- Effacez le volume du catalogue avec la commande **delete volume** dans la console (sélectionnez le VolumeName ou le MediaId lorsque vous y êtes invité).
- Utilisez la commande **unmount** pour démonter l'ancienne cartouche.
- Ré-étiquetez physiquement l'ancienne cartouche de sorte qu'elle puisse être réutilisée.
- Insérez l'ancienne cartouche dans le lecteur.
- Depuis la ligne de commande, saisissez : **mt -f /dev/st0 rewind** et **mt -f /dev/st0 weof**, où vous prendrez soin de substituer la chaîne désignant votre lecteur à **/dev/st0**.
- Utilisez la commande **label** dans la console pour écrire une nouvelle étiquette Bacula sur votre cartouche.
- Utilisez la commande **mount**, si ce n'est pas réalisé automatiquement, afin que Bacula commence à utiliser la cartouche fraichement étiquetée.

D'autres commandes utiles de la console Bacula

status dir Affiche un état de tous les jobs en cours d'exécution ainsi que tous les jobs programmés dans les prochine 24 heures

status Le programme Console vous invite à sélectionner un *daemon*, puis il s'enquiert de l'état de ce *daemon*.

status jobid=nn Affiche un état du JobId nn s'il est en cours d'exécution. Le Storage Daemon est aussi contacté pour produire un état du job.

list pools Affiche la liste des pools définis dans le catalogue.

list media Affiche la liste des média définis dans le catalogue.

list jobs Affiche la liste de tous les jobs enregistrés dans le catalogue et squi ont été exécutés.

list jobid=nn Affiche le JobId nn depuis le catalogue.

list jobtotals Affiche les totaux pour tous le jobs du catalogue.

list files jobid=nn Affiche la liste des fichiers sauvegardés pour le JobId nn.

list jobmedia Affiche des informations relatives aux média utilisés pour chaque job exécuté.

messages Affiche tous les messages redirigés vers la console.

unmount storage=storage-name Démonte le lecteur associé au périphérique de stockage désigné par **storage-name** s'il n'est pas en cours d'utilisation. Cette commande est utile si vous souhaitez que Bacula libère le lecteur.

mount storage=storage-name Le lecteur associé au périphérique de stockage est monté à nouveau. Lorsque Bacula atteint la fin d'un volume et vous demande d'en monter un nouveau, vous devez utiliser cette commande après avoir introduit une nouvelle cartouche dans le lecteur. En effet, c'est le signal qui indique à Bacula qu'il peut commencer à lire ou écrire sur la cartouche.

quit Permet de quitter le programme Console.

La plupart des commandes citées ci-dessus, à l'exception de **list**, vous invitent à compléter la liste des arguments fournis si vous vous contentez d'entrer le nom de la commande.

Débugger la sortie des daemons

Si vous voulez débugger la sortie des *daemons* en cours d'exécution, lancez-les, depuis le répertoire d'installation, comme suit :

```
./bacula start -d100
```

Cette possibilité peut vous fournir une aide précieuse si vos *daemons* ne démarrent pas correctement. Normalement, la sortie des *daemons* est dirigée vers le périphérique NULL, avec un niveau de débuggage supérieur à zéro, elle est dirigée vers le terminal de lancement.
Pour stopper les trois *daemons*, tapez simplement :

```
./bacula stop
```

dans le répertoire d'installation.
L'exécution de **bacula stop** peut signaler des pids non trouvés. C'est Ok, spécialement si l'un des **bacula stop** est mort, ce qui est très rare.
Pour faire une sauvegarde complète (Full) du système, chaque File Daemon doit être exécuté en tant que root afin d'avoir les permissions requises pour

accéder à tous les fichiers. Les autres *daemons* n'ont pas besoin des privilèges root. Cependant, le Storage Daemon doit être capable d'accéder aux lecteurs, ce qui Sur beaucoup de systèmes, n'est possible que pour root. Vous pouvez, au choix, exécuter le Storage Daemon en tant que root, ou changer les permissions sur les lecteurs pour autoriser les accès non-root. MySQL et PostgreSQL peuvent être installés et exécutés avec un userid quelconque, les privilèges root ne sont pas requis.

Soyez patient lorsque vous démarrez les *daemons* ou montez des cartouches vierges

Lorsque vous lancez les *daemons* Bacula, le Storage Daemon tente d'ouvrir tous les périphériques de stockage définis et de vérifier le volumes courrament montés. Il n'accepte aucune connection de la console tant que tous les périphériques n'ont pas été vérifiés. Une cartouche qui a été utilisé précédemment doit être rembobinée, ce qui, sur certain lecteurs, peut prendre plusieurs minutes. Par conséquent, vous devriez faire preuve d'un peu de patience lorsue vous tentez de contacter le Storage Daemon pour la première fois après le lancement de Bacula. Si vous avez un accès visuel à votre lecteur, celui-ci devrait être prèt à l'emploi lorsque son témoin lumineux cesse de clignoter.
Les mêmes considérations s'appliquent si vous avez monté une cartouche vierge dans un lecteur tels qu'un HP DLT. Il peut s'écouler une à deux minutes avant que le lecteur se rende compte que la cartouche est vierge. Si vous tentez de la monter pendant cette période, il est probable que vous aller geler votre pilote SCSI (c'est le cas sur mon système RedHat). Par conséquent, nous vous enjoignons une fois encore à être patient lors de l'insertion de cartouches vierges. Laissez le lecteur s'initialiser avant de tenter d'y accéder.

Problèmes de connection du FD vers le SD

Si l'un ou plusieurs de vos File Daemons rencontre des difficultés à se connecter au Storage Daemon, c'est très probablement que vous n'avez pas utilisé un nom pleinement qualifié pour la directive **Address** de la ressource Storage du fichier de configuration du Director. Le résolveur de la machine cliente (celle qui exécute le FD) doit être capable de résoudre le nom que vous avez spécifié dans cette directive en une adresse IP. Un exemple d'adresse ne fonctionnant pas est **localhost**. Un exemple qui pourrait fonctionner : **megalon**. Un exemple qui a encore plus de chances de fonctionner : **magalon.mydomain.com**. Sur les systèmes Win32, si vous ne disposez pas d'un bon résolveur (c'est souvent le cas sur Win98), vous pouvez essayer en utilisant une adresse IP plutôt qu'un nom.

Si votre adresse est correcte, assurez vous qu'aucun autre programme n'utilise le port 9103 sur la machine qui héberge le Storage Daemon. Les numéros de ports de Bacula sont autorisés par l'IANA, et ne devraient donc pas être utilisés par d'autres programmes, mais il semble que certaines imprimantes HP les utilisent. Exécutez la commande **netstat -a** sur la machine qui héberge le Storage Daemon pour déterminer qui utilise le port 9103 (utilisé pour les communications du FD vers le SD).

Options en ligne de commande des Daemons

Chacun des trois *daemons* (Director, File, Storage) acceptent quelques options sur la ligne de commande. En général, chacun d'entre eux, de même que le programme Console, admet les otpions suivantes :

-c <file> Définit le fichier de configuration à utiliser. La valeur par défaut est le nom du *daemon* suivi de **conf**, par exemple **bacula -dir.conf** pour le Director, **bacula-fd** pour le File Daemon, et **bacula-sd.conf** pour le Storage Daemon.

-d nn Fixe le niveau de débuggage à la valeur **nn**. Les niveaux les plus élevé permettent d'afficher plus d'information sur STDOUT concernant ce que le *daemon* est en train de faire.

-f Exécute le *daemon* en arrière plan. Cette option est requise pour exécuter les *daemon*s avec le debugger.

-s Ne pas capturer les signaux. Cette option est requise pour exécuter les *daemon*s avec le debugger.

-t Lire les fichiers de configuration et afficher les messages d'erreur, et quitter immédiatement. Très utile pour tester la syntaxe de nouveaux fichiers de configuration.

-v Mode verbeux. Utile pour rendre les messages d'erreur et d'information plus complets.

-? Affiche la version et la liste des options.

Le Director a les options spécifiques suivantes :

-r <job> Exécute le job désigné immédiatement. Ceci ne devrait servir qu'à des fins de débuggage.

Le File Daemon les options spécifiques suivantes :

-i Suppose que le *daemon* est appelé par **inetd** ou **xinetd**. Dans ce cas, le *daemon* suppose qu'une connection est déjà établie et qu'elle est passée en tant que STDIN. Le *daemon* s'arrète dès que la connection se termine.

Le Storage Daemon n'a pas d'options spécifiques.
Le programme Console n'a pas d'options spécifiques.

Créer un Pool

La création de pool est automatique au démarrage de Bacula, aussi si vous comprenez déjà le concept de pools et leur fonctionnement, vous pouvez passer à la section suivante.
Lorsque vous exécutez un job, Bacula doit déterminer quel volume utiliser pour sauvegarder le FileSet. Plutôt que de spécifier un volume directement, vous spécifiez l'ensemble de volumes dans lequel vous autorisez Bacula à puiser lorsqu'il lui faut un volume pour écrire les données sauvegardées.Dès lors, Bacula se charge de sélectionner le premier volume utilisable dans le pool approprié au périphérique que vous avez spécifié pour le job exécuté. Lorsqu'un volume est plein, Bacula change son VolStatus de **Append** en **Full**, et utilise le volume suivant, et ainsi de de suite. S'il n'y a pas de volume utilisable, Bacula envoie un message à l'opérateur pour réclamer la création d'un volume approprié.
Bacula garde trace des noms de pools, des volumes contenus dans les pools, et de plusieurs caractéristiques de chacun de ces volumes.
Lorsque Bacula démarre, il s'assure que toutes les définitions de ressources Pool ont été enregistrées dans le catalogue. Vous pouvez le vérifier avec la commande :

```
list pools
```

du programme Console, qui devrait produire quelque chose comme :

```
*list pools
Using default Catalog name=MySQL DB=bacula
+--------+---------+---------+---------+----------+-------------+
| PoolId | Name    | NumVols | MaxVols | PoolType | LabelFormat |
+--------+---------+---------+---------+----------+-------------+
| 1      | Default | 3       | 0       | Backup   | *           |
| 2      | File    | 12      | 12      | Backup   | File        |
+--------+---------+---------+---------+----------+-------------+
*
```

Si vous tentez de créer un pool existant, Bacula affiche :

```
Error: Pool Default already exists.
Once created, you may use the {\bf update} command to
modify many of the values in the Pool record.
```

Etiqueter vos Volumes

Bacula exige que chaque volume comporte une étiquette (NDT : label) logicielle. Il existe plusieurs stratégies pour étiqueter les volumes. Celle que j'utilise consiste à les étiqueter à l'aide du programme Console au fur et à mesure qu'ils sont requis par Bacula. Ainsi, lorsqu'il a besoin d'un volume qu'il ne trouve pas dans son catalogue, Bacula m'envoie un e-mail pour

m'enjoindre à ajouter un volume au pool. J'utilise alors la commande **label** dans la console pour étiqueter un nouveau volume et le définir dans le catalogue, après quoi Bacula est en mesure de l'utiliser. Alternativement, je peux utiliser la commande **relabel** pour ré-étiquter un volume qui n'est plus utilisé, pourvu qu'il ait le VolStatus **Purged**.
Une autre stratégie consiste à étiqueter un ensemble de volumes, et à les utiliser au fur et à mesure que Bacula les réclame. C'est le plus souvent ce qui est fait lorsque vous cyclez sur un groupe de volumes, par exemple avec une librairie. Pour plus de détails sur le recyclage, veuillez consulter le chapitre Recyclage automatique des volumes de ce manuel.
Si vous exécutez un job Bacula alors que vous n'avez pas de volumes étiquetés dans le pool concerné, Bacula vous en informe, et vous pouvez les créer "à la volée". Dans mon cas, j'étiquette mes cartouches avec la date, par exemple : **DLT-18April02**. Voyez ci-dessous pour plus de détails sur l'usage de la commande **label**.

Etiquetage des volumes dans la console

L'étiquetage des volumes se fait, en principe, avec le programme Console.

1. ./bconsole
2. label

Si Bacula annonce que vous ne pouvez étiqueter une cartouche au motif qu'elle porte déjà une étiquette, démontez-la avec la commande **unmount**, puis recommencez avec une cartouche vierge.
Etand donné que le support de stockage physique est différent pour chaque périphérique, la commande **label** vous propose une liste de ressources Storage définies telle que celle-ci :

```
The defined Storage resources are:
     1: File
     2: 8mmDrive
     3: DLTDrive
     4: SDT-10000
Select Storage resource (1-4):
```

A ce stade, vous devriez avoir une cartouche vierge dans votre lecteur d'un type correspondant à la ressource Storage que vous avez sélectionné.
Bacula vous demande le nom du volume :

```
Enter new Volume name:
```

S'il proteste :

```
Media record for Volume xxxx already exists.
```

Cela signifie que le nom de volume **xxxx** que vous avez entré existe dèjà dans le catalogue. Vous pouvez afficher la liste des média définis avec la commande **list media**. Notez que la colonne LastWritten a ici été tronquée pour permettre un affichage propre.

```
+---------------+---------+--------+----------------+-----/~/-+------------+-----+
| VolumeName    | MediaTyp| VolStat| VolBytes       | LastWri | VolReten   | Recy|
+---------------+---------+--------+----------------+---------+------------+-----+
| DLTVol0002    | DLT8000 | Purged | 56,128,042,217 | 2001-10 | 31,536,000 |   0 |
| DLT-07Oct2001 | DLT8000 | Full   | 56,172,030,586 | 2001-11 | 31,536,000 |   0 |
| DLT-08Nov2001 | DLT8000 | Full   | 55,691,684,216 | 2001-12 | 31,536,000 |   0 |
| DLT-01Dec2001 | DLT8000 | Full   | 55,162,215,866 | 2001-12 | 31,536,000 |   0 |
| DLT-28Dec2001 | DLT8000 | Full   | 57,888,007,042 | 2002-01 | 31,536,000 |   0 |
| DLT-20Jan2002 | DLT8000 | Full   | 57,003,507,308 | 2002-02 | 31,536,000 |   0 |
| DLT-16Feb2002 | DLT8000 | Full   | 55,772,630,824 | 2002-03 | 31,536,000 |   0 |
| DLT-12Mar2002 | DLT8000 | Full   | 50,666,320,453 | 1970-01 | 31,536,000 |   0 |
| DLT-27Mar2002 | DLT8000 | Full   | 57,592,952,309 | 2002-04 | 31,536,000 |   0 |
| DLT-15Apr2002 | DLT8000 | Full   | 57,190,864,185 | 2002-05 | 31,536,000 |   0 |
| DLT-04May2002 | DLT8000 | Full   | 60,486,677,724 | 2002-05 | 31,536,000 |   0 |
| DLT-26May02   | DLT8000 | Append |  1,336,699,620 | 2002-05 | 31,536,000 |   1 |
+---------------+---------+--------+----------------+-----/~/-+------------+-----+
```

Une fois que Bacula a vérifié que le volume n'existe pas encore, il vous demande le pool dans lequel vous souhaitez que le volume soit créé. S'il n'existe qu'un pool, il est sélectionné automatiquement.

Si la cartouche est étiquetée correctement, un enregistrement de volume est aussi créé dans le pool. Ainsi, le nom du volume et tous ses attributs apparaîtront lorque vous afficherez les volumes du pool. De plus, le volume est disponible pour les sauvegardes, pourvu que le MediaType coincide avec celui requis par le Storage Daemon.

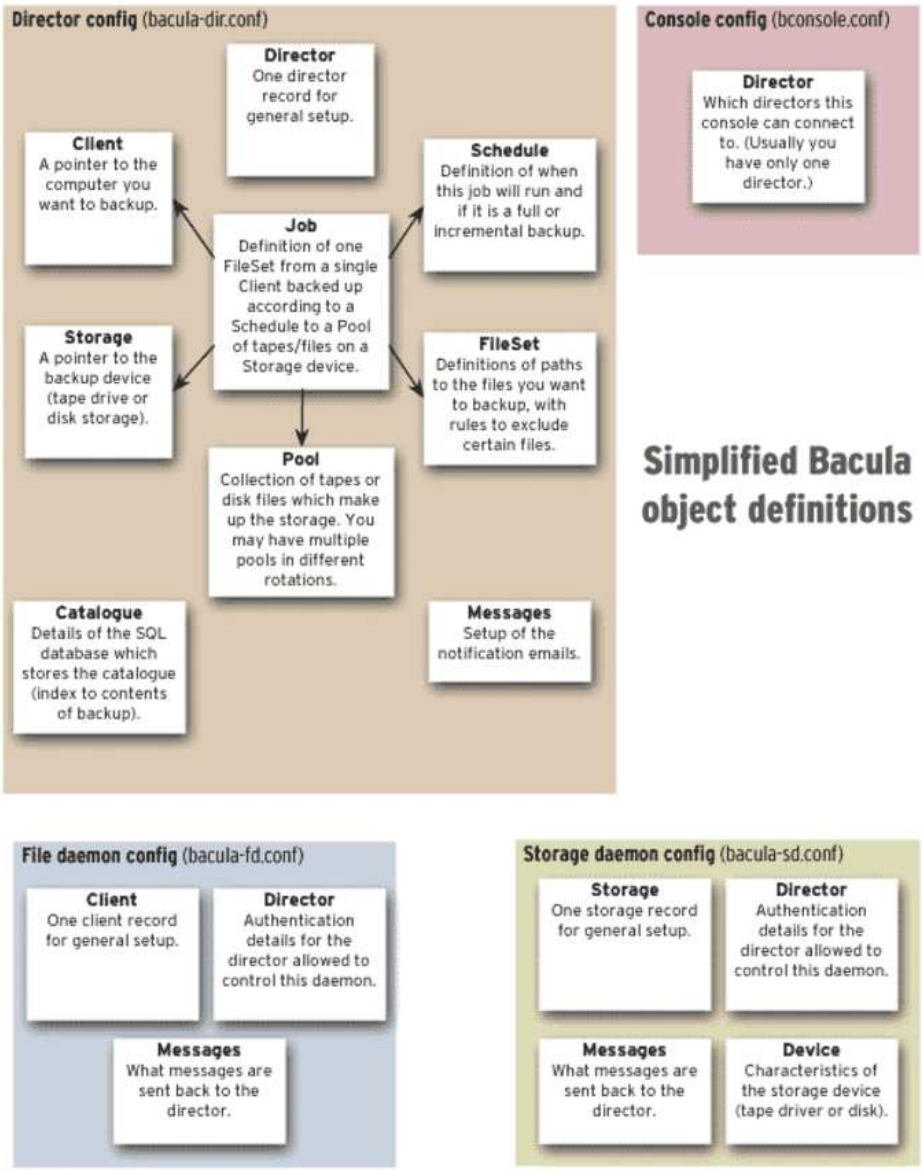
Lorsque vous avez étiqueté la cartouche, vous n'avez répondu qu'à quelques questions la concernant – principalement son nom, et éventuellement le *Slot*. Cependant, un enregistrement de volume dans le catalogue (connu au niveau interne en tant qu'enregistrement Media) contient un certain nombre d'attributs. La plupart d'entre eux sont renseignés selon les valeurs par défaut qui ont été définies lors de la création du pool (au trement dit, le pool comporte la plupart des attributs par défaut utilisés lors de la création d'un volume). Il est aussi possible d'ajouter des media aux pools sans les étiqueter physiquement. C'est la fonction de la commande **add**. Pour plus d'informations, veuillez consulterle chapitre Console de ce manuel.

Adapter les fichiers de configuration

Lors de son démarrage, chacun des programmes qui composent Bacula lit un fichier de configuration spécifié sur la ligne de commande, ou par défaut **bacula-dir.conf**, **bacula-fd.conf**, **bacula-sd.conf**, ou **console.conf** pour le Director Daemon, le File Daemon, le Storage Daemon, et le programme Console respectivement.

Chaque service (Director, Client, Storage, Console) possède son propre fichier de configuration qui contient un groupe de directives. Dans la suite, nous désignerons ces groupes de directives par le mot **Ressource**. Les ressources son très similaires d'un service à l'autre, mais peuvent contenir des directives différentes selon les services. Par exemple, dans le fichier de configuration du Director, la ressource **Director** définit le nom du Director, quelques paramètres généraux du Director et son mot de passe. Dans le fichier de configuration du File Daemon, la ressource **Director** spécifie les Directors autorisés à utiliser le File Daemon.

Avant de lancer Bacula pour la première fois, vous devez adapter chaque fichier de configuration. Des fichiers de configuration auront été créés par le processus d'installation, mais il doivent être modifiés pour correspondre à votre système. Le schéma suivant donne une vue globale des ressources.



(Remerciements à Aristedes Maniatis pour ce schéma.)

Format des directives

Bien qu'il ne soit pas nécessaire de connaître le détail de toutes les directives possibles, une connaissance basique des ressources Bacula est indispensable. Chaque directive contenue dans une ressource (entre accollades) est composé d'un mot-clef suivi du signe "=", suivi d'une ou plusieurs valeurs. Le mot clef doit être l'un de ceux connus par Bacula, et peut comporter des caractères majuscules et minuscules ainsi que des espaces.

Chaque définition de ressource DOIT comporter la directive Name, et peut optionnellement comporter la directive Description. La directive Name est utilisée pour identifier de façon unique la ressource. La directive Description sera utilisé lors de l'affichage pour offrir une identification plus aisée de la ressource. Par exemple :

```
Director {
  Name = "MyDir"
  Description = "Main Bacula Director"
  WorkingDirectory = "$HOME/bacula/bin/working"
}
```

Définit la ressource Director avec le nom "MyDir" et le répertoire de travail $HOME/bacula/bin/working. En général, si vous voulez utiliser des espaces dans le nom à droite du signe "=", vous devez l'entourer de doubles quotes. Sinon, les quotes ne sont généralement pas requises car une fois définies, les chaînes quotées et non quotées sont toutes équivalentes.

Commentaires

Lors de la lecture d'un fichier de configuration, les lignes blanches sont ignorées, et tout ce qui suit le caractère dièse (#) jusqu'à la fin de la ligne est considéré comme commentaire. Un point virgule (;) indique la fin logique d'une ligne et tout ce qui suit le point virgule est considéré comme le paramètre suivant. Un paramètre qui apparaît seul sur une ligne ne nécessite pas de point virgule, vous ne verrez donc pas beaucoup de points virgule dans les exemples de ce manuel.

Casse et espaces

La casse (majuscules/minuscules) et les espaces sont totalement ignorées dans les mots-clef des directives des ressources (la partie à gauche du signe "=").
A l'intérieur des mots-clef (à gauche du signe "="), les espaces ne sont pas significatives. Ainsi, les mots-clef : **name**, **Name**, et **N a m e** sont tous identiques.
Les espaces après le signe "=" et avant le premier caractère de la valeur son ignorées.
En général, les espaces à l'intérieur d'une valeur sont significatives (non ignorées), et si la valeur est un nom, vous devez l'encadrer de doubles quotes pour que l'espace soit acceptée. Les noms peuvent contenir jusqu'à 127 caractères. Actuellement, un nom peut contenir n'importe quel caractère ASCII. A l'intérieur d'une chaîne quotée, tout caractère précédé d'un backslash (\) est pris tel quel (utile pour insérer des backslashes et doubles quotes (")).
Veuillez cependant noter que les noms de ressource Bacula ainsi que certains autres noms seront sévèrement limités dans l'avenir pour ne plus autoriser

que les lettres (y compris accentuées ISO), nombres, et une poignée de caractères spéciaux (espaces, underscores, ...). Tous les autres caratères et ponctuations seront prohibés.

Inclure d'autres fichiers de configuration

Si vous souhaitez éclater votre fichier de configuration en fichiers plus petits, l'inclusion est possible avec la syntaxe @**NomDeFichier** où **nom de fichier** est le chemin absolu vers un le fichier à inclure. Toute donnée primitive peut être remplacée par une spécification @NomDeFichier. Par exemple
Director { Name=@xxx est valide et substituera le fichier xxx à @xxx, mais
Director { Name@xxx = something n'est pas valide puisque @xxx apparaît au milieu d'un (???mot clef/directive/...???)

Types de données primitives reconnus

Lorsqu'il parcourt les enregistrements de ressource, Bacula classe les données selon les types enumérés ci-dessous. En première lecture, cette liste peut vous paraître accablante, mais en réalité, tout y est d'une logique élégante et directe.

name Un mot-clef ou un nom constitué de caractères alpha-numériques, incluant le trait d'union, underscore et dollar. Le premier caractère d'un **name** doit être une lettre. La longueur d'un **name** est actuellement limitée à 127 caractères. Typiquement, les mots-clef apparaîssent à gauche d'un signe "=". Les mots-clef ne peuvent être quotés.

name-string Une **name-string** est similaire à un **name** excepté qu'elle peut être quotée et ainsi contenir des caractères supplémentaires. La longueur des **name-strings** est limitée à 127 caractères. Typiquement, on les utilise à droite d'un signe "=", (i.e. ce sont les valeurs à associer aux mots-clef).

string Une chaîne quotée contenant potentiellement n'importe quel caractère, y compris les espaces, ou une chaîne non quotée. Une **string** peut avoir une longueur quelconque. Typiquement, les **strings** sont les valeurs qui correspondent aux noms de fichiers, répertoires, ou noms de commandes système. Un *Backslash* (\) change le prochain caractère en lui même, de sorte que pour inclure une double quote dans une chaîne, vous devez la préceder d'un *Backslash*. Il en va de même pour inclure un *Backslash*.

directory Un **directory** (Répertoire) est une chaîne, quotée ou non. Un **directory** est transmis à votre shell standard pour expansion lorsqu'il est scanné. Ainsi, des constructions telles que **$HOME** sont interprétées correctement.

password Il s'agit d'un mot de passe Bacula. Il est stocké en interne sous un format brouillé par MD5.

integer Une valeur entière relative (positive ou négative) sur 32 bits.

positive integer Une valeur entière positive sur 32 bits.

long integer Une valeur entière sur 64 bits. Typiquement, ce sont les valeurs telles que le nombre de bytes, qui peuvent dépasser 4 milliards et ainsi nécessitent une valeur sur 64 bits.

yes—no Soit **yes**, soit **no**.

size Une taille specifiée en bytes. Typiquement, il s'agit d'une entrée au format scientifique (avec virgule flottante) suivie d'un modificateur optionnel. L'entrée est stockée en tant qu'entier sur 64 bits. Si un modificateur est spécifié, il doit suivre immédiatement la valeur, sans espace. Les modificateurs suivants sont reconnus:

k 1 024 (kilobytes)
kb 1 000 (kilobytes)
m 1 048 576 (megabytes)
mb 1 000 000 (megabytes)
g 1 073 741 824 (gigabytes)
gb 1 000 000 000 (gigabytes)

time Une heure, ou une durée spécifiée en secondes. Une valeur **time** est stockée en interne en tant qu'entier sur 64 bits, mais il est spécifié en deux parties: un nombre et un modificateur. Le nombre peut être un entier ou un nombre à virgule flottante. S'il est spécifié en virgule flottante, il sera arrondi à l'entier le plus proche. Le modificateur est obligatoire et suit le nombre avec ou sans espace intercalée. Les modificateurs suivants sont reconnus:

seconds secondes
minutes minutes (60 secondes)
hours heures (3600 secondes)
days jours (3600*24 secondes)
weeks semaines (3600*24*7 secondes)
months mois (3600*24*30 secondes)
quarters trimestres (3600*24*91 secondes)
years années (3600*24*365 secondes)

Toute abbréviation des ces modificateurs est aussi autorisée (i.e. **seconds** peut être spécifié par **sec** ou **s**. Une spécification **m** sera interprétée en tant que mois.

La spécification d'une durée peut comporter autant de couples nombre/modificateur que vous le souhaitez. Par exemple:

```
1 week 2 days 3 hours 10 mins
1 month 2 days 30 sec
```

sont des spécifications valides (à partir de la version 1.35.1).
Note! Dans les versions de Bacula antérieures à 1.31, le modificateur était optionnel. Il est désormais obligatoire.

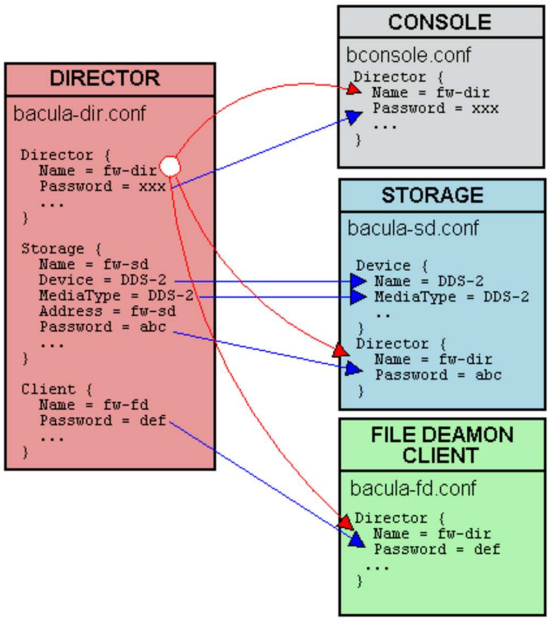
Types de Ressources

La table suivante énumère tous les types de ressource de la version courante de Bacula. Elle montre quelle ressource doit être définie pour chaque service (*daemon*). Les fichiers de configuration par défaut contiennent au moins un exemple de chaque ressource permise, aussi ne soyez pas angoissé à l'idée d'avoir à créer toutes ces directives *ex nihilo*.

Resource	**Director**	**Client**	**Storage**	**Console**
Autochanger	No	No	Yes	No
Catalog	Oui	Non	Non	Non
Client	Oui	Oui	Non	Non
Console	Oui	Non	Non	Oui
Device	Non	Non	Oui	Non
Director	Oui	Oui	Oui	Oui
FileSet	Oui	Non	Non	Non
Job	Oui	Non	Non	Non
JobDefs	Oui	Non	Non	Non
Message	Oui	Oui	Oui	Non
Pool	Oui	Non	Non	Non
Schedule	Oui	Non	Non	Non
Storage	Oui	Non	Oui	Non

Noms, mots de passe et autorisations

Pour qu'un *daemon* puisse en contacter un autre, il lui faut s'authentifier avec un mot de passe. Dans la plupart des cas, le mot de passe est associé à un nom particulier, de sorte que nom et mot de passe doivent correspondre. Les fichiers de configuration par défaut sont automatiquement définis avec des autorisations correctes et des mots de passe aléatoires. Si vous modifiez ces fichiers, vous devrez être attentif à leur cohérence.
Voici un schéma des correspondances que doivent respecter les couples "nom/mot de passe" des différents fichiers de configurations.



Dans la colonne de gauche, vous trouverez les ressources Director, Storage et Client, avec leurs noms et mots de passe – ils sont tous dans le fichier **bacula-dir.conf**. Dans la colonne de droite figurent les valeurs correspondantes dans les fichiers de configuration de la Console, du Storage Daemon (SD) et du File Daemon (FD).

Veuillez noter que l'adresse **fd-sd**, qui apparaît dans la ressource Storage du Director, précédée d'une atérisque est transmise au File Daemon sous forme symbolique. Le File Daemon la résout alors en une adresse IP. Pour cette raison, vous devez utiliser soit une adresse IP, soit une adresse pleinement qualifiée. Une adresse telle que **localhost**, n'étant pas pleinement qualifiée, sera résolue par le File Daemon en l'hôte local du File Daemon, ce qui n'est probablement pas le résultat désiré. Le mot de passe utilisé par le File Daemon pour autoriser la communication avec le Storage Daemon est temporaire et unique pour chaque *job*. Il n'est spécifié dans aucun fichier de configuration.

Informations détaillées sur chaque *daemon*

Les détails de chaque ressource et des directives permises sont décrits dans les chapitres suivants.

Les fichiers de configuration suivants doivent être définis:

- Console – Pour définir les ressources pour le programme Console (interface utilisateur avec le Director). Ce fichier définit quels sont les Directors disponibles avec lesquels vous pouvez interagir.

– Director – Pour définir les ressources nécessaires au Director. Vous y définissez tous les Clients et Storage Daemons que vous utilisez.
– Client – Pour définir les ressources de chaque client à sauvegarder. Vous aurez un fichier de ressources Client pour chacune des machine qui exécute un File Daemon.
– Storage – Pour définir les ressources utilisées par chaque Storage Daemon. En principe, vous aurez un seul Storage Daemon qui contrôlera votre (vos) lecteur(s). Quoi qu'il en soit, si vous avez des lecteurs sur plusieurs machines, vous aurez au moins un Storage Daemon par machine.

Configurer le Director

Parmi tous les fichiers de configuration requis pour exécuter **Bacula**, celui du Director est le plus compliqué, et c'est celui que vous modifierez le plus souvent, en ajoutant des clients ou en modifiant les FileSets.
Pour une discussion générale concernant les fichiers et ressources ainsi que les types de données reconnus par **Bacula**, veuillez consulter le chapitre Configuration de ce manuel.

Les types de ressources du Director

Les types de ressources du Director sont :
Job, JobDefs, Client, Storage, Catalog, Schedule, FileSet, Pool, Director, et Messages. Nous les présentons ici dans l'ordre le plus logique (relativement au fichier de configuration du Director) :

- Director – Pour définir le nom du Director et son mot de passe pour l'authentification du programme Console. Il ne doit y avoir qu'une seule définition de ressource Director dans le fichier de configuration. Si vous avez soit **/dev/random** soit **bc** sur votre machine, Bacula génèrera un mot de passe aléatoire lors du processus de configuration, sinon, il sera laissé blanc.
- Job – Pour définir les Jobs de types sauvegarde et restauration, et pour lier les ressources Client, FileSet et Schedules à utiliser conjointement pour chaque Job.
- JobDefs – Ressource optionnelle destinée à fournir des valeurs par défaut pour les ressources Job.
- Schedule – Pour définir le moment où un Job doit être lancé automatiquement par le *scheduler* interne de Bacula.
- FileSet – Pour définir l'ensemble des fichiers à sauvegarder pour chaque client.
- Client – Pour définir quel Client est à sauvegarder.
- Storage – Pour définir sur quel périphérique physique les volumes seront montés.
- Pool – Pour définir quel le pool de volumes qui peut être utilisé pour un Job donné
- Catalog – Pour définir la base de données où conserver les listes des fichiers sauvegardés et des volumes où ils ont été sauvegardés.
- Messages – Pour définir les destinataires (ou les fichiers de logs) des messages d'erreur et d'information.

La ressource Director

La ressource Director définit les attributs du Director exécuté sur le réseau. Dans l'implémentation actuelle, il n'y a qu'une ressource Director, mais la réalisation finale contiendra plusieurs Directors pour maintenir la redondance de la base des indexes et média.

Director Début de la ressource Director. Une ressource Director et une seule doit être définie.

Name = <nom> Le nom du Director utilisé par l'administrateur système. Cette directive est requise

Description = < texte> Le champ texte contient une description du Director qui sera affichée dans l'interface graphique. Cette directive est optionnelle.

Password = <UA-password> Spécifie le mot de passe qui doit être fourni par la Console Bacula par défaut pour être autorisée. Le même mot de passe doit apparaître dans la ressource **Director** du fichier de configuration de la console. Pour plus de sécurité, le mot de passe ne transite jamais sur le réseau, l'authentification se fait via un échange de type *challenge-response* d'un *hash code* créé avec le mot de passe. Cette directive est requise. Si vous disposez de **/dev/random** ou **bc** sur votre machine, Bacula génèrera un mot de passe aléatoire lors du processus d'installation, sinon il sera laissé blanc et vous devrez en définir un manuellement.

Messages = <Nom-de-ressource-Messages> La ressource **messages** spécifie où doivent être délivrés les messages du Director qui ne sont pas associés à un job spécifique. La plupart des messages sont relatifs à un job et seront dirigés vers la ressource **messages** spécifiée par le job. Cependant, quelques messages peuvent être générés lorsque aucun job n'est actif. Cette directive est requise.

Working Directory = <Répertoire> Cette directive spécifie un répertoire où le Director peut déposer ses fichiers d'états. Ce répertoire ne devrait être utilisé que par Bacula, mais il peut être partagé par d'autres *daemons* Bacula. Notez cependant que si ce répertoire est partagé avec d'autres daemons Bacula, vous devez vous assurer que le nom **Name** donné à chaque daemon est unique afin d'éviter des collisions entre les fichiers temporaires utilisés. Par défaut, le processus de configuration de Bacula créé des noms de daemons uniques en les postfixant avec -dir, -fd et -sd. Les substitutions shell standard sont effectuées à la lecture du fichier de configuration, de sorte que des valeurs telles que **$HOME** seront correctement substituées. Cette directive est requise.

Si vous avez spécifié un utilisateur et/ou un groupe pour le Director lors de la configuration avec les options **--with-dir-user** et/ou **--with-dir-group** de la commande ./configure, le répertoire de travail Working Directory doit appartenir doit appartenir à ce groupe et à cet utilisateur.

Pid Directory = <Répertoire> Cette directive spécifie un répertoire où le Director peut déposer son fichier d'Id de processus. Ce fichier est utilisé pour stopper Bacula et prévenir l'exécution simultanée de plusieurs copies de Bacula. Les substitutions shell standard sont effectuées à la lecture du fichier de configuration, de sorte que des valeurs telles que **$HOME** seront correctement substituées.

Typiquement, sur les systèmes Linux, vous utiliserez ici **/var/run**. Si vous n'installez pas Bacula dans les répertoires système, vous pouvez utiliser le répertoire de travail **Working Directory** défini plus haut. Cette directive est requise.

Scripts Directory = <Directory> Cette directive spécifie le répertoire dans lequel le Director devra rechercher le script Python de démarrage **DirStartup.py**. Ce répertoire peut être partagé par d'autres daemons Bacula.Les substitutions shell standard sont effectuées à la lecture du fichier de configuration, de sorte que des valeurs telles que **$HOME** seront correctement substituées. Cette directive est optionnelle.

QueryFile = <Path> Cette directive spécifie un répertoire et un fichier dans lequel le Director peut trouver les requêtes SQL préétablies pour la commande **Query** de la Console. Les substitutions shell standard sont effectuées à la lecture du fichier de configuration, de sorte que des valeurs telles que **$HOME** seront correctement substituées. Cette directive est requise.

Maximum Concurrent Jobs = <nombre> Où <nombre> est le nombre maximal de jobs qui peuvent être exécutés simultanément par le Director. La valeur par défaut est 1, mais vous pouvez utiliser une valeur plus grande. Notez que le format des volumes devient beaucoup plus compliqué avec plusieurs jobs exécutés simultanément. De ce fait, les restaurations peuvent prendre beaucoup plus de temps si Bacula doit faire le tri parmi les segments entremélés de ces jobs. Ceci peut être évité en s'arrangeant pour que chacun des jobs exécutés simultanément écrive sur un volume distinct. Une autre possibilité consiste à utiliser le *data spooling* : les données seront d'abord "spoolées" sur disque simultanément, ensuite les fichiers "spool" seront écrits séquentiellement sur le volume.

Dans certains cas, des directives telles que **Maximum Volume**

Jobs ne sont pas correctement synchronisées avec le nombre de jobs simultanés, et des problèmes de synchronisation subtils peuvent survenir, aussi des tests minutieux sont recommandés.

Actuellement, il n'y a aucun paramètre de configuration pour régler ou limiter le nombre de connections par console. Un maximum de cinq connection simultanées est autorisé.

Pour plus de détails concernant l'exécution simultanée de plusieurs jobs, consultez la partie Exécution simultanée de plusieurs jobs du chapitre Astuces de ce manuel.

FD Connect Timeout = <durée> Où **durée** est le délai durant lequel le Director tente de contacter le File Daemon pour démarrer un job. Une fois ce délai écoulé, le Director supprimera le job. La valeur par défaut est 30 minutes.

SD Connect Timeout = <durée> Où **durée** est le délai durant lequel le Director tente de contacter le Storage Daemon pour démarrer un job. Une fois ce délai écoulé, le Director supprimera le job. La valeur par défaut est 30 minutes.

DirAddresses = <Spécification-adresses-IP> Spécifie les ports et adresses sur lesquels le Director se met en attente de connections de Consoles Bacula. La meilleure explication est sans doute un exemple :

```
DirAddresses  = { ip = {
       addr = 1.2.3.4; port = 1205;}
   ipv4 = {
       addr = 1.2.3.4; port = http;}
   ipv6 = {
       addr = 1.2.3.4;
       port = 1205;
   }
   ip = {
       addr = 1.2.3.4
       port = 1205
   }
   ip = {
       addr = 1.2.3.4
   }
   ip = {
       addr = 201:220:222::2
   }
   ip = {
       addr = bluedot.thun.net
   }
}
```

où "ip", "ip4", "ip6", "addr", et "port sont les mots clef. Notez que les adresses peuvent être spécifiées sous forme de quadruplets

pointés, ou suivant la notation à doubles points IPv6, ou encore sous forme de nom symbolique (seulement pour la spécification ip). D'autre part, le port peut être spécifié par un nombre, ou par une valeur mnémonique du fichier /etc/services. Si un port n'est pas précisé, celui par défaut sera utilisé. Si une section ip est spécifiée, la résolution peut être faite soit par IPv4, soit par IPv6. Si ip4 est spécifié, seules les résolutions IPv4 seront permises. Il en va de même avec ip6.

Notez que si vous utilisez la directive DirAddresses, vous ne devez utiliser ni la directive DirPort, ni la directive DirAddress dans la même ressource.

DIRport = <numéro-de-port> Spécifie le port (un entier positif) sur lequel le Director est à l'écoute de connections de Consoles Bacula. Ce même numéro de port doit être spécifié dans la ressource Director du fichier de configuration de la console. La valeur par défaut est 9101, aussi, il n'est en principe pas nécessaire de renseigner cette directive. Elle n'est pas requise si vous spécifiez des DirAdresses.

DirAddress = <Adresse-IP> Cette directive est optionnelle. Lorsqu'elle est spécifiée, le Director n'accepte de connections Console que de l'adresse spécifiée **Adresse-IP**, qui peut être soit un nom de domaine, soit une adresse IP au format quadruplet pointé ou chaîne quotée. Si cette directive n'est pas spécifiée, le Director acceptera des connections de Console de toute adresse valide. Notez que contrairement à la spécification DirAdresses décrite plus haut, cette directive ne permet de spécifier qu'une seule adresse. Cette directive n'est pas requise si vous utilisez la directive DirAdresses.

Voici un exemple d'une ressource Director valide :

```
Director {
  Name = HeadMan
  WorkingDirectory = "$HOME/bacula/bin/working"
  Password = UA_password
  PidDirectory = "$HOME/bacula/bin/working"
  QueryFile = "$HOME/bacula/bin/query.sql"
  Messages = Standard
}
```

La ressource Job

La ressource Job définit un Job (sauvegarde, restauration,...) que Bacula doit exécuter. Chaque définition de ressource Job contient le nom d'un client, la liste des éléments à sauvegarder (FileSet), la planification (Schedule) pour ce Job, le lieu où sauvegarder ces données (Storage Device) et quel groupe

de media utiliser (Pool). En effet, chaque ressource Job doit répondre aux questions : "Quoi ?", "Où ?", "Quand ?" et "Comment ?" soit, respectivement Fileset, Storage, Schedule, Type et Niveau (Sauvegarde/Restauration - Full/Differentielle/Incrémentale). Notez que le FileSet doit être spécifié lors des restaurations pour des raisons historiques, mais il n'est plus utilisé. Un seul type (**Backup**, **Restore**, ...) peut être spécifié pour un Job donné. Si vous voulez sauvegarder plusieurs FileSets sur le même client, vous devez définir un Job pour chacun d'entre eux.

Job Début de la ressource Job. Il faut définir au moins une ressource Job.

Name = <name> Le nom du Job. Ce nom peut être utilisé avec la commande **Run** du programme Console pour lancer un Job. Si le nom contient des espaces, il doit être placé entre quotes. C'est généralement une bonne idée de nommer vos Jobs du nom du Client qu'ils sauvegardent, afin de les identifier aisément.

Lors de l'exécution d'un Job, son nom unique est composé du nom que vous avez spécifié ici suffixé avec la date et l'heure de sa planification. Cette directive est requise.

Enabled = <yes—no> irective!Enable Cette directive permet d'activer ou désactiver l'exécution automatique d'un job par le planificateur.

Type = <job-type> La directive **Type** spécifie le type de Job, qui peut être l'un des suivants : **Backup**, **Restore**, **Verify**, ou **Admin**. Cette directive est requise. Pour chaque type de Job, il existe différents niveaux, qui seront décrits dans les prochains paragraphes.

Backup Définit une sauvegarde. En principe, vous aurez au moins un job de type Backup par client sauvegardé. A moins que vous ne désactiviez le catalogue, la plupart des données et statistiques concernant les fichiers sauvegardées seront écrites dans le catalogue.

Restore Définit une restauration. En principe, vous ne créerez qu'un seul job de ce type, que vous utiliserez comme un prototype que vous modifierez à l'aide de la console lorsque vous devrez restaurer. Bien que certaines informations basiques soient conservées dans le catalogue lors de restaurations, leur quantité est infime en regard des informations stockées pour une sauvegarde – par exemple, aucune entrée de nom de fichier n'est générée, puisqu'aucun fichier n'est sauvegardé.

Verify Définit un Job de type Verify. Le Jobs de type **verify** permettent de comparer le catalogue au système de fichiers ou à ce qui a été sauvegardé. Vous pouvez l'utiliser pour

vous assurer qu'une cartouche de données est lisible, mais aussi comme un système de détection d'intrusion à la fade Tripwire.

Admin Définit un Job de type Admin. Un **Admin** peut s'utiliser pour "élaguer" périodiquement le catalogue, si vous ne souhaitez pas que ceci soit fait à la fin de chaque sauvegarde. Bien que les Jobs de type admin soient enregistrés dans le catalogue, la quantité de données générée est infime.

Level = <job-level> La directive Level spécifie le niveau d'exécutiondu job par défaut. Chaque type de job a son propre jeu de niveaux qui peuvent être spécifiés. Le niveau d'exécution est en général surchargé par une valeur différente spécifiée dans la ressource **Schedule**. Cette directive n'est pas requise mais doit être spécifiée soit ici, soit en tant que surcharge dans la ressource **Schedule**.

Pour un job de type **Backup** le niveau doit être l'un des suivants :

Full Tous les fichiers du FileSet, qu'ils aient été modifiés ou non.

Incremental Tous les fichiers modifiés depuis la dernière sauvegarde valide du FileSet spécifié pour le même job. Si le Director ne peut trouver une sauvegarde Full antérieure, le niveau du job sera élevé en une sauvegarde Full. Lorsque le Director recherche une Full valide dans le catalogue, il recherche un job avec les caractéristiques suivantes :

- le même nom de job ;
- le même nom de client ;
- le même FileSet (toute modification de la définition du FileSet telle que l'ajout ou la suppression de fichiers dans les sections Include ou Exclude constitue un changement de FileSet).
- le niveau requis (Full, Differential ou Incremental)
- le job s'est terminé normalement, c'est à dire un qu'il ne s'est pas terminé en échec, et n'a pas été effacé.

Si toutes les conditions ci-dessus ne sont pas réalisées, le Director augmentera la sauvegarde incrémentale en une sauvegarde Full. Dans le cas contraire, la sauvegarde incrémentale sera effectuée normalement.

Le File Daemon (Client) détermine les fichiers à sauvegarder pour une incrémentale par comparaison de l'heure de démarrage du Job précédent (Full, Différentiel ou Incrémental) avec les dates de dernière modification de

chaque fichier (st_mtime) et de ses attributs (st_ctime). Si le fichier ou ses attributs ont changés depuis cette date de démarrage, alors le fichier sera sauvegardé.
Veuillez noter que certains logiciels anti-virus peuvent modifier la date st_time lors de leurs opérations de scan. Ainsi, si l'antivirus modifie la date d'accès (st_atime), qui n'est pas utilisée par Bacula, il provoquera une modification du st_ctime et conduira Bacula à sauvegarder les fichiers concernés lors des incrémentales et différentielles. Dans le cas de l'antivirus Sophos, vous pouvez éviter cet inconvénient en utilisant l'option **--no-reset-atime**. Pour les autres logiciels, voyez leurs manuels.
Lorsque Bacula effectue une sauvegarde incrémentale, tous les fichiers modifiés présents sur le système sont sauvegardés. Cependant, tout fichier supprimé depuis la dernière Full demeure dans le catalogue, ce qui signifie que si vous effectuez une restauration à partir de sauvegardes incrémentales (et de la Full associée), les fichiers supprimés depuis la dernière Full seront aussi restaurés. Ces fichiers n'apparaîtront plus dans le catalogue après avoir fait une nouvelle sauvegarde Full. Le processus pour supprimer ces fichiers du catalogue lors d'une incrémentale ralentirait fortement les sauvegardes incrémentales. Il n'est actuellement pas implémenté dans Bacula.
De plus, si vous déplacez un répertoire plutôt que de le copier, les fichiers qu'il contient voient leurs dates de dernière modification (st_mtime) et de dernier accès (st_ctime) inchangés. Par conséquent, ces fichiers ne seront probablement sauvegardés par aucune incrémentale ou différentielle, puisque ces dernières ne se réfèrent qu'à ces indicateurs. Aussi, il est préférable de copier un dossier avant de supprimer l'original plutôt que de le déplacer, si vous voulez qu'il soit correctement sauvegardé.

Differential Tous les fichiers modifiés depuis la dernière sauvegarde Full valide du FileSet spécifié pour le même job. Si le Director ne peut trouver une sauvegarde Full antérieure, le niveau du job sera élevé en une sauvegarde Full. Lorsque le Director recherche une Full valide dans le catalogue, il recherche un job avec les caractéristiques suivantes :

- le même nom de job ;
- le même nom de client ;
- le même FileSet (toute modification de la définition du FileSet telle que l'ajout ou la suppression de fichiers dans

les sections Include ou Exclude constitue un changement de FileSet).
- le Job était une sauvegarde FULL
- le Job s'est terminé normalement, c'est à dire qu'il ne s'est pas terminé en échec, et n'a pas été effacé.

Si toutes les conditions ci-dessus ne sont pas réalisées, le Director augmentera la sauvegarde différentielle en une sauvegarde Full. Dans le cas contraire, la sauvegarde différentielle sera effectuée normalement.
Le File Daemon (Client) détermine les fichiers à sauvegarder pour une différentielle par comparaison de l'heure de démarrage de la dernière sauvegarde Full avec les dates de dernière modification de chaque fichier (st_mtime) et de ses attributs (st_ctime). Si le fichier ou ses attributs ont changés depuis cette date de démarrage, alors le fichier sera sauvegardé. La date de démarrage utilisée est affiché après le **Since** du rapport de Job. Dans de rares cas, certains fichiers sont sauvegardés deux fois à cause de l'utilisation de la date de démarrage de la sauvegarde précédente, mais ceci assure qu'aucun changement n'est perdu. Comme pour les incrémentales, vous devriez vous assurer que les horloges de votre serveur Bacula et de vos clients sont synchronisées, ou aussi proches que possible, pour éviter le risque d'omission d'un fichier. Notez qu'à partir de la version 1.33, Bacula effectue automatiquement ces ajustements de sorte que les horloges utilisées par Bacula soient synchrones.
Veuillez noter que certains logiciels anti-virus peuvent modifier la date st_time lors de leurs opérations de scan. Ainsi, si l'antivirus modifie la date d'accès (st_atime), qui n'est pas utilisée par Bacula, il provoquera une modification du st_ctime et conduira Bacula à sauvegarder les fichiers concernés lors des incrémentales et différentielles. Dans le cas de l'antivirus Sophos, vous pouvez éviter cet inconvénient en utilisant l'option **--no-reset-atime**. Pour les autres logiciels, voyez leurs manuels.
Lorsque Bacula effectue une sauvegarde différentielle, tous les fichiers modifiés présents sur le système sont sauvegardés. Cependant, tout fichier supprimé depuis la dernière Full demeure dans le catalogue, ce qui signifie que si vous effectuez une restauration à partir de sauvegardes différentielles (et de la Full associée), les fichiers supprimés depuis la dernière Full seront aussi restaurés. Ces fichiers n'apparaîtront plus dans le catalogue après avoir fait une nouvelle sauvegarde

Full. Le processus pour supprimer ces fichiers du catalogue lors d'une incrémentale ralentirait fortement les sauvegardes différentielles. Il n'est actuellement pas implémenté dans Bacula, mais plannifié pour une future version de Bacula.

Comme noté ci-dessus, si vous déplacez un répertoire plutôt que de le copier, les fichiers qu'il contient voient leurs dates de dernière modification (st_mtime) et de dernier accès (st_ctime) inchangés. Par conséquent, ces fichiers ne seront probablement sauvegardés par aucune incrémentale ou différentielle, puisque ces dernières ne se réfèrent qu'à ces indicateurs. Aussi, il est préférable de copier un dossier avant de supprimer l'original plutôt que de le déplacer, si vous voulez qu'il soit correctement sauvegardé.

Régulièrement, quelqu'un demande à quoi servent les sauvegardes différentielles du moment que les incrémentales récupèrent tous les fichiers modifiés. Il existe plusieurs réponses à cette question, mais la plus importante à mes yeux est de combiner toutes les incrémentales et différentielles depuis la dernière full en une seule différentielle. Ceci a deux effets : 1. La redondance. 2. Plus important, la réduction du nombre de volumes requis pour faire une restauration en éliminant la nécessité de lire tous les volumes des précédentes incrémentales depuis la dernière full.

Pour un Job de type **Restore**, aucun niveau ne doit être spécifié.
Pour un Job de type **Verify**, le niveau peut être l'un des suivants :

InitCatalog Examine le **FileSet** spécifié et stocke les attributs de fichiers dans le catalogue. Vous pouvez vous interroger sur l'intérêt d'un Job qui ne sauvegarde aucun fichier. La réponse est de pouvoir utiliser Bacula comme vous utiliseriez Tripwire, en d'autres termes, ce type de Jobs vous permet de sauvegarder l'état d'un ensemble de fichiers défini par un **FileSet** afin de pouvoir ultérieurement contrôler si rien n'a été modifié, supprimé ou ajouté. Ceci peut être utilisé pour détecter une intrusion. Typiquement, vous spécifiez un **FileSet** qui contient l'ensemble des fichiers qui ne devraient pas changer (par exemple /sbin, /boot, /lib, /bin, ...). Ensuite, vous exécutez le Job verify de niveau **InitCatalog** après l'installation de votre système, puis après chaque modification (mise à jour). Ensuite, lorsque vous souhaitez contrôler l'état de votre système de fichiers, vous utilisez un Job **Verify**, **level = Catalog** afin de comparer le résultat de votre **InitCatalog** avec l'état actuel de votre système de fichiers.

Catalog Compare l'état actuel des fichiers et l'état précédemment sauvegardé lors d'un **InitCatalog**. Toutes les anomalies sont rapportées. Les objets du rapport sont déterminés par les options **verify** spécifiées dans la directive **Include** du **FileSet** spécifié (voyez la ressource **FileSet** ci-dessous pour plus de détails). Typiquement, cette commande sera exécutée quotidiennement pour contrôler toute modification de votre système de fichier.

Attention ! Si vous exécutez deux jobs Verify Catalog simultanément sur le même client, les résultats seront probablement erronnés. En effet, Verify Catalog modifie le catalogue lors de son exécution afin de détecter les nouveaux fichiers.

VolumeToCatalog Ce niveau permet de lire les attributs de fichiers écrits sur le Volume lors du dernier Job. Les attributs de fichiers sont comparés aux valeurs sauvegardées dans le catalogue et toute différence est rapportée. Ceci est similaire au niveau **Catalog**, sauf que ce sont les attributs des fichiers du volume plutôt que ceux des fichiers du disque qui sont comparés aux attributs sauvegardés dans le catalogue. Bien que les attributs et signatures (MD5 ou SHA1) soient comparés, les données réelles ne le sont pas (elles ne figurent pas dans le catalogue).

Attention ! Si vous exécutez deux jobs Verify VolumeToCatalog simultanément sur le même client, les résultats seront probablement erronnés. En effet, Verify VolumeToCatalog modifie le catalogue lors de son exécution afin de détecter les nouveaux fichiers.

DiskToCatalog Ce niveau permet de lire les fichiers tels qu'ils sont actuellement sur le disque et de comparer leurs attributs actuels avec ceux stockés dans le catalogue lors de la dernière sauvegarde pour le Job spécifié par la directive **VerifyJob**. Ce niveau diffère du niveau **Catalog** décrit plus haut en ce qu'il ne se réfère pas à un Job Verify antérieur, mais à la dernière sauvegarde. Lorsque vous utilisez ce niveau , vous devez renseigner les option Verify de la section Include. Ces options déterminent quels attributs seront comparés.

Cette commande peut se révéler très utile si vous avez des problèmes de disque car elle comparera l'état actuel de votre disque avec la dernière sauvegarde valide, qui peut remonter à plusieurs jobs.

Notez que l'implémentation actuelle (1.32c) n'identifie pas les fichiers qui ont été supprimés.

Verify Job = <Job-Resource-Name> Si vous exécutez un job

verify sans cette directive, le dernier job exécuté sera comparé avec le catalogue, ce qui signifie que votre commande verify doit succéder immédiatement à une sauvegarde. Si vous spécifiez un **Verify Job**, Bacula trouvera le dernier job exécuté avec ce nom. Ceci vous permet d'exécuter toutes vos sauvegardes, puis d'exécuter les jobs Verify sur les sauvegardes de votre choix (le plus souvent, un **VolumeToCatalog** de sorte que la cartouche qui vient juste d'être écrite est relue).

JobDefs = <JobDefs-Resource-Name> Si un nom de JobDef est spécifié dans la définition d'un Job, toutes les valeurs définies dans la ressource JobDef concernée seront utilisées en tant que valeurs par défaut pour le Job. Toute valeur explicitement spécifiée dans la définition du Job outrepasse la valeur par défaut définie par le JobDef. L'utilisation de cette directive permet d'écrire des ressources Job plus compactes, où la majeure partie des directives sont définies dans un ou plusieurs JobDefs. C'est particulièrement pratique si vous avez de nombreux Jobs similaires avec des variations mineures telles que différents clients. Un exemple basique de l'utilisation d'un Jobdef est fourni dans le fichier bacula-dir.conf par défaut.

Bootstrap = <bootstrap-file> La directive Bootstrap spécifie un fichier bootstrap qui, s'il est fourni, sera utilisé lors des restaurations et ignoré par tout les autres Jobs. Le fichier **bootstrap** contient la liste des cartouches nécessaires pour la restauration ainsi que les index des fichiers à restaurer (localisation sur la cartouche). Cette directive est optionnelle, et n'est utilisée que pour les restaurations. De plus, elle peut être modifiée lorsqu'une restauration est lancée depuis la console.

Si vous utilisez la commande **Restore** dans la console pour lancer une restauration, le fichier **bootstrap** sera créé automatiquement à partir des fichiers que vous avez sélectionnés pour la restauration.

Pour plus de détails concernant les fichiers **bootstrap**, veuillez consulter le chapitre Restaurer des fichiers avec le fichier Bootstrap de ce manuel.

Write Bootstrap = <bootstrap-file-specification> La directive **writebootstrap** spécifie le de fichier Bootstrap où Bacula écrira lors de chaque sauvegarde. Ainsi, cette directive s'applique exclusivement aux jobs de type sauvegarde. Si la sauvegarde est une Full, Bacula écrase le contenu du fichier spécifié. Sinon, Bacula ajoute les nouveaux enregistrements Bootstrap à la fin du fichier.

En utilisant cette fonction, vous aurez constamment un fichier

bootstrap capable de recouvrer l'état le plus récent de votre système. Le fichier bootstrap devrait être écrit sur un disque monté sur une autre machine, de sorte que vous puissiez en disposer immédiatement en cas de défaillance de votre disque dur. Une alternative consiste à copier le fichier sur une autre machine après chaque mise à jour.

Si la **spécification de fichier bootstrap** débute par une barre verticale (—), Bacula considère la spécification comme un nom de programme vers lequel les les enregistrement bootstrap seront redirigés. Ce peut être, par exemple, un script qui vous envoie par e-mail les enregistrements bootstrap.

Pour plus de détails sur l'utilisation de fichiers bootstrap, veuillez consulter le chapitre intitulé Le Fichier Bootstrap de ce manuel.

Client = <client-resource-name> La directive Client spécifie le Client (File Daemon) à utiliser dans le Job. Le client est exécuté sur la machine à sauvegarder. Il expédie les fichiers requis au Storage Daemon lors des sauvegardes, et reoit les fichiers du Storage Daemon lors des restaurations. Pour plus de détails, consultez la section Ressource Client de ce chapitre. Cette deirective est requise.

FileSet = <FileSet-resource-name> La directive FileSet spécifie le FileSet à utiliser dans le Job concerné. Le FileSet définit les répertoires et fichiers à sauvegarder, ainsi que les options à utiliser pour les sauvegarder (par exemple la compression,...). Un Job ne peut contenir qu'un seul FileSet. Pour plus de détails, consultez la section Ressource FileSet de ce chapitre. Cette directive est requise.

Messages = <messages-resource-name> La directive Messages définit la ressource Message qui doit être utilisée pour le job concerné. Ainsi, elle détermine le comment et où seront délivrés les différents messages de Bacula. Par exemple, vous pouvez diriger certains messages vers un fichier de logs, tandis que d'autres seront envoyés par e-mail. Pour plus de détails, consultez le chapitre Ressource Messages de ce manuel. Cette directive est requise.

Pool = <pool-resource-name> La directive Pool spécifie le jeu de volumes qui doit être utilisé pour sauvegarder vos données. De nombreuses installations de Bacula n'utiliseront que le pool défini par défaut **Default**. Toutefois, si vous voulez spécifier différents jeux de volumes pou différents clients ou différents jobs, vous voudrez probablement utiliser les Pools. Pour plus de détails, consultez la section Ressource Pool de ce chapitre. Cette directive est requise

Full Backup Pool = <pool-resource-name> La directive *Full Backup Pool* spécifie un Pool à utiliser pour les sauvegardes Full. Cette directive outrepasse toute autre spécification de Pool lors d'une sauvegarde Full. Cette directive est optionnelle.

Differential Backup Pool = <pool-resource-name> La directive *Differential Backup Pool* spécifie un Pool à utiliser pour les sauvegardes Différentielles. Cette directive outrepasse toute autre spécification de Pool lors d'une sauvegarde Différentielle. Cette directive est optionnelle.

Incremental Backup Pool = <pool-resource-name> La directive *Incremental Backup Pool* spécifie un Pool à utiliser pour les sauvegardes Incrémentales. Cette directive outrepasse toute autre spécification de Pool lors d'une sauvegarde Incrémentale. Cette directive est optionnelle.

Schedule = <schedule-name> La directive Schedule définit la planification du job. Le *schedule* détermine la date et l'instant où le job doit être lancé automatiquement, et le niveau (Full, Différentiel, Incrémental...) du job en question. Cette directive est optionnelle. Si elle est omise, le job ne pourra être exécuté que manuellement via la Console. Bien que vous puissiez vous contenter d'une ressource Schedule simple pour tout job, vous pouvez aussi définir des ressources Schedule avec plusieurs directives **Run**, afin de lancer le job à différentes heures. Chacune de ces directives **Run** permet d'outrepasser les valeurs par défaut de Level, Pool, Storage et Messages ressources. Ceci autorise une grande souplesse d'utilisation d'un simple job. Pour plus de détails, consultez le chapitre. Ressource Schedule de ce manuel.

Storage = <storage-resource-name> La directive Storage définit le nom du service storage que vous souhaitez utiliser pour sauvegarder les données du FileSet. Pour plus de détails, consultez le chapitre Ressource Storage de ce manuel. Cette directive est requise.

Max Start Delay = <time> La directive Max Start Delay spécifie le délai maximal entre l'horaire planifié (dans le schedule) et l'horaire effectif de démarrage du job. Par exemple, un job peut être programmé pour démarrer à 1h, mais être mis en attente à cause d'autres jobs en cours d'exécution. Si le Max Start Delay a été réglé à 3600, le job sera supprimmé s'il n'a pas démarré à 2h. Ceci peut se révéler utile pour, par exemple, éviter qu'un job s'exécute duant les heures ouvrables. La valeur par défaut est 0 (pas de limite).

Max Run Time = <time> La directive Max Run Time spécifie le

délai alloué pour l'exécution complète d'un job depuis son lancement (pas nécessairement à l'heure de sa programmation) jusqu'à sa fin. Cette directive est implémentée depuis la version 1.33.

Max Wait Time = <time> La directive Max Wait Time spécifie le délai maximum durant lequel un job peut rester bloqué en attente d'une ressource (par exemple, en attente du montage d'une cartouche ou encore en attente des Storage ou File Daemon occupés à d'autres tches) depuis son lancement (pas nécessairement à l'heure de sa programmation) jusqu'à sa fin. Cette directive est implémentée depuis la version 1.33.

Incremental Max Wait Time = <time> Cette directive spécifie le temps maximum durant lequel une sauvegarde incrémentale peut rester bloquée en attente d'une ressource (par exemple, en attente d'une cartouche ou des File ou Storage daemons), compté à partir du démarrage du job, (**pas nécessairement** l'heure à laquelle le job a été programmé). Notez que si un **Max Wait Time** a été spécifié, il peut aussi s'appliquer au job.

Differential Max Wait Time = <time> Cette directive spécifie le temps maximum durant lequel une sauvegarde différentielle peut rester bloquée en attente d'une ressource (par exemple, en attente d'une cartouche ou des File ou Storage daemons), compté à partir du démarrage du job, (**pas nécessairement** l'heure à laquelle le job a été programmé). Notez que si un **Max Wait Time** a été spécifié, il peut aussi s'appliquer au job.

Prefer Mounted Volumes = <yes—no> Si cette directive est activée (c'est le cas par défaut), le Director ordonne au Storage daemon de sélectionner de préférence soit une librairie, soit un lecteur avec un volume valide déjà monté, plutôt qu'un lecteur pas prèt. Si aucun lecteur n'est prèt, c'est le premier lecteur prèt qui sera sélectionné.

Si cette directive est désactivée, le Storage daemon privilégiera les lecteurs inutilisés. Ce mode de fonctionnement peut être très utile pour ces sites avec de nombreux lecteurs qui où il peut être préférable de maximiser le flux des sauvegardes au prix d'une utilisation d'un plus grand nombre de lecteurs et de cartouches. Afin d'optimiser l'utilisation de plusieurs lecteurs, vous voudrez probablement lancer chacun de vos jobs l'un après l'autre avec un intervalle de 5 secondes environ. Ceci aidera à assurer que chaque nuit, le même lecteur (volume) est sélectionné pour le même job. Autrement, lors d'une restauration, vous pourriez trouver vos fichiers dispersés sur beaucoup plus de volumes que nécessaire.

Prune Jobs = <yes—no> En principe, l'élagage des jobs du catalogue est spécifié pour chaque client dans sa propre ressource

Client par la directive **AutoPrune**. Si cette directive est spécifiée (normalement, non) et si la valeur est **yes**, elle outrepasse la valeur spécifiée dans la ressource Client. La valeur par défaut est **no**.

Prune Files = <yes—no> En principe, l'élagage des fichiers du catalogue est spécifié pour chaque client dans sa propre ressource Client par la directive **AutoPrune**. Si cette directive est spécifiée (normalement, non) et si la valeur est **yes**, elle outrepasse la valeur spécifiée dans la ressource Client. La valeur par défaut est **no**.

Prune Volumes = <yes—no> En principe, l'élagage des volumes du catalogue est spécifié pour chaque client dans sa propre ressource Client par la directive **AutoPrune**. Si cette directive est spécifiée (normalement, non) et si la valeur est **yes**, elle outrepasse la valeur spécifiée dans la ressource Client. La valeur par défaut est **no**.

Run Before Job = <command> La commande spécifiée est exécutée en tant que programme externe avant le lancement du job. Tout retour de la commande sur la sortie standard est incluse dans le rapport de job de Bacula. La chaîne bf command doit être un nom de programme valide ou un script shell. Cette directive n'est pas requise, mais si elle est définie, et si le code retour d'exécution du programme est différent de zéro, le job qui a lancé le programme est effacé. D'autre part, la chaîne bf command est parcourue puis envoyée vers la fonction execvp(), ce qui signifie que le chemin de la commande est recherché pour son exécution, mais qu'il n'y a aucune interprétation shell. Par conséquent, si vous voulez utiliser des commandes complexes ou toute fonctionnalité du shell telle que la redirection, vous devez appeler un script shell où vous mettrez vos commandes. Avant de soumettre la commande spécifiée au système d'exploitation, Bacula effectue les substitutions suivantes :

```
%% = %
%c = Nom du client
%d = Nom du Director

%e = Statut de sortie du job
%i = JobId
%j = Nom unique du job
%l = Niveau du job
%n = Nom du job
%s = Temps Depuis (NDT : Since Time)
%t = Type de job (Backup,...)
%v = Nom de volume
```

Le le code de statut de fin de job peut prendre les valeurs suivantes :

- OK
- Error
- Fatal Error
- Canceled
- Differences
- Unknown term code

Aussi, si vous l'utilisez dans une ligne de commande, il vous faudra l'encadrer de quotes.
Depuis la version 1.30, Bacula contrôle le statut de sortie du programme RunBeforeJob. S'il est différent de zéro, le job se termine en erreur. Lutz Kittler a fait remarquer que ceci peut être un moyen aisé pour modifier vos schedules pour les vacances. Par exemple, supposons que vous fassiez habituellement des sauvegardes Full le vendredi, mais que jeudi et vendredi soient fériés. Pour éviter d'avoir à changer les cartouches entre jeudi et vendredi alors que personne n'est au bureau, vous pouvez créer un RunBeforeJob qui retourne un statut non nul jeudi et zéro les autres jours. Ainsi, le job de jeudi ne sera pas exécuté, et la cartouche que vous avez inséré mercredi sera disponible pour la Full de vendredi.

Run After Job = <command> La commande spécifiée est exécutée en tant que programme externe après la fin du job. La chaîne bf command doit être un nom de programme valide ou un script shell. Cette directive n'est pas requise. Si le code de sortie du programme est non nul, Bacula affiche un message d'avertissement (warning). Avant de soumettre la commande spécifiée au système d'exploitation, Bacula effectue les substitutions de caractères décrites au paragraphe **Run Before Job**
Un exemple d'utilisation de cette directive est donné au chapitre Astuces de ce manuel. Lisez le paragraphe **Run After Failed Job** si vous voulez exécuter une commande lorqu'un job se termine avec un statut anormal.

Run After Failed Job = <command> La commande spécifiée est exécutée en tant que programme externe après la fin du job lorqu'il se termine avec un statut d'erreur. Cette directive est optionnelle. La chaîne **command** doit être le nom d'un programme ou d'un script shell. Si le code de sortie du programme est non nul, Bacula affiche un mesage d'avertissement (warning). Avant de soumettre la commande spécifiée au système d'exploitation,

Bacula effectue les substitutions de caractères décrites au paragraphe **Run Before Job**. Notez que vous pouvez, si vous le souhaitez, spécifier ici le même programme que celui que vous avez utilisé pour la directive **Run After Job**, de sorte que votre programme soit exécuté quel que soit l'issue du job.

Le chapitre Trucs et astuces de ce manuel propose un exemple d'utilisation de cette directive.

Client Run Before Job = <command> Cette directive est similaire à **Run Before Job** excepté que la commande est exécutée sur la machine cliente. Les mêmes restrictions s'appliquent aux sytèmes Unix que celles signalées pour **Run Before Job**.

Lorsque vous spécifiez un chemin absolu vers un exécutable, si le chemin ou le nom de l'exécutable contient des espaces ou des caractères spéciaux, il faut les protéger par des quotes. Il en va de même des éventuels arguments.

Considérations particulières à Windows D'autre part, pour les clients Windows à partir de la version 1.33, notez bien que vous devez fournir un chemin correct pour votre script, et que le script peut avoir l'extension .com, .exe, ou .bat. Si vous spécifiez un chemin, vous devez aussi spécifier l'extension complète. Les commandes à la façon d'Unix ne fonctionneront pas, à moins que vous n'ayez installé et correctement configuré Cygwin en plus (et séparément) de Bacula.

La commande peut être n'importe quel programme reconnu par cmd.exe ou command.com comme un fichier exécutable. Spécifier une extension de fichier exécutable est optionnel, à moins qu'il y ait une ambiguïté (par exemple ls.bat, ls.exe).

Bacula cherche la commande dans le répertoire "System %Path%" (Dans la boîte de dialogue des variables d'environnement vous avez les variables "système" et "utilisateurs". Si bacula-fd fonctionne en tant que service, seules les variables d'environnement systèmes sont accessibles.)

Les variables d'environnement système peuvent être invoquées avec la syntaxe %var% et utilisées comme portion du nom de la commande ou des arguments.

Lorsque la spécification du chemin absolu d'un exécutable ou le nom de l'exécutable contient des espaces ou des caractères spéciaux, ils doivent être quotés. Il en va de même pour les arguments.

```
ClientRunBeforeJob = "\"C:/Program Files/Software
     Vendor/Executable\" /arg1 /arg2 \"foo bar\""
```

Les caractères spéciaux &()[]{}^=;!'+,‘~ devront être quotés s'ils font partie d'un nom de fichier ou d'un argument.

If someone is logged in a blank “command” window running the commands will be present during the execution of the command. Quelques suggestions de Phil Stracchino pour l'exécution sur les machines Win32 avec le File Daemon Win32 natif :

1. Vous pourriez utiliser la directive ClientRunBeforeJob pour spécifier un fichier .bat qui exécute les commandes coté client plutôt que d'essayer d'exécuter (par exemple) regedit /e directement.
2. Le fichier batch devrait retourner explicitement 0 lors des exécutions correctes.
3. Le chemin vers le fichier batch devrait être spécifié au format Unix :
   ClientRunBeforeJob = “c:/bacula/bin/systemstate.bat”
   plutôt qu'au format DOS/Windows :
   ClientRunBeforeJob = “c:\bacula\bin\systemstate.bat” INCORRECT

L'exemple suivant d'utilisation de la directive Client Run Before Job a été soumis par un utilisateur :
Vous pourriez écrire un script shell pour sauvegarder une base DB2 dans un FIFO. Voici le script en question :

```
#!/bin/sh
# ===== backupdb.sh
DIR=/u01/mercuryd

mkfifo $DIR/dbpipe
db2 BACKUP DATABASE mercuryd TO $DIR/dbpipe WITHOUT PROMPTING &
sleep 1
```

La ligne suivante dans la ressource Job du fichier bacula-dir.conf :

```
Client Run Before Job = "su - mercuryd -c \"/u01/mercuryd/backupdb.sh '%t' '%l'\""
```

Lorsque le job est exécuté, vous obtiendrez un message de sortie du script annoant que la sauvegarde a démarré. Même si la commande est exécutée en arrière plan avec &, le job bloquera jusqu'à la commande "db2 BACKUP DATABASE", et la sauvegarde se fige. Pour remédier à cette situation, la ligne "db2 BACKUP DATABASE" devrait être modifiée en :

```
db2 BACKUP DATABASE mercuryd TO $DIR/dbpipe WITHOUT PROMPTING > $DIR/backup.log 2>&1 < /dev/null &
```

Il est important de rediriger l'entrée et la sortie d'une commande en arrière plan vers /dev/null pour éviter le bloquage du script.

Client Run After Job = <command> Cette directive est similaire à **Run After Job** sauf qu'elle est exécutée sur la machine cliente. Veuillez consulter les notes concernant les clients Windows dans le paragraphe **Client Run Before Job** ci-dessus.

Rerun Failed Levels = <yes—no> Si la valeur de cette directive est **yes** (**no** par défaut), et si Bacula détecte qu'un job antérieur d'un niveau plus élevé (Full ou différentiel), alors le job est élevé au niveau le plus haut. Ceci est particulièrement utile pour sauvegarder les pc portables qui peuvent être fréquemment inaccessibles. En effet, après l'échec d'une Full, vous souhaiterez probablement que la prochaine sauvegarde soit de niveau Full plutôt qu'Incremental ou Différentiel.

Spool Data = <yes—no> Si la valeur de cette directive est **yes** (**no** par défaut), le Storage Daemon aura pour consigne de stocker les données dans un fichier spoule sur disque plutôt que de les écrire directement sur bande. Lorsque toutes les données sont dans le spoule ou lorsque la taille maximale fixée pour le fichier spoule est atteinte, les données sont déchargées du spoule vers les bandes. Lorsque la valeur de cette directive est **yes**, la directive Spool Attributes est aussi automatiquement mise à la valeur **yes**. L'utilisation de cette fonctionnalité prévient les arrêts et redémarrage incessants lors des incrémentales. Elle ne doit pas être utilisée si vous sauvegardez sur disque.

Spool Attributes = <yes—no> La valeur par défaut est **no**, ce qui signifie que le Storage Daemon envoie les attributs de fichiers au Director au moment où ils (les fichiers) sont ecrits sur la bande. Cependant, si vous souhaitez éviter le risque de ralentissement dû aux mises à jour du catalogue, vous pouvez régler cette directive à **yes**, dans ce cas, le Storage Daemon stockera les attributs de fichiers dans un fichier tampon du Working Directory pour ne les transmettre au Director qu'à la fin de l'écriture sur bande des données du job.

Where = <directory> Cette directive ne concerne que les jobs de type Restauration. Elle permet de spécifier un préfixe au nom du répertoire où tous les fichiers sont restaurés. Ceci permet de restaurer les fichiers en un emplacement différent de celui où ils ont été sauvegardés. Si **Where** n'est pas renseigné, ou si sa valeur est backslash (/), les fichiers sont restaurés à leur emplacement d'origine. Par défaut, nous avons donné à **Where** la valeur **/tmp/bacula-restores** dans les fichiers de configuration fournis en exemple, ceci afin d'éviter l'écrasement accidentel de vos fichiers.

Replace = <replace-option> Cette directive ne concerne que les jobs de type Restauration. Elle précise la conduite à adopter dans l'éventualité où Bacula serait conduit à restaurer un fichier ou un répertoire qui existe déjà. Les options suivantes sont disponibles :

always Lorsque le fichier à restaurer existe déjà, il est supprimé et remplacé par la copie sauvegardée.

ifnewer Si le fichier sauvegardé (sur bande) est plus récent que le fichier existant, le fichier existant est supprimé et remplacé par la copie sauvegardée.

ifolder Si le fichier sauvegardé (sur bande) est plus ancien que le fichier existant, le fichier existant est supprimé et remplacé par la copie sauvegardée.

never Si le fichier sauvegardé existe déjà, Bacula renonce à restaurer ce fichier.

Prefix Links=<yes—no> Si la valeur de cette directive est **Yes** et si un préfixe de chemin **Where** est spécifié, alors ce dernier s'applique aussi aux liens absolus. La valeur par défaut est **No**. Lorsque cette directive est à **Yes**, tous les liens absolus seront aussi modifiés pour pointer vers le nouveau répertoire. En principe, c'est ce qui est souhaité : l'ensemble du répertoire restauré conserve sa cohérence interne. Cependant, si vous voulez replacer les fichiers ultérieurement à leurs emplacements d'origine, tous les liens absolus seront brisés.

Maximum Concurrent Jobs = <number> Où <number> est le nombre maximum de jobs de la ressource Job courrante qui peuvent être exécutés simultanément. Notez que cette directive ne limite que les jobs avec le même nom que la ressource dans laquelle elle figure. Toute autre restriction du nombre maximum de jobs simultanés, que ce soit au niveau du Director, du Client ou de la ressource Storage, s'applique en plus de de la limite stipulée ici. LA valeur par défaut est 1, mais vous pouvez utiliser une valeur plus grande. Nous vous recommandons fortement de lire attentivement le paragraphe WARNING sous Maximum Concurrent Jobs dans la section concernant la ressource Director.

Reschedule On Error = <yes—no> Si cette directive est activée, alors si le job se termine en erreur, il sera reprogrammé en accord avec les directives **Reschedule Interval** et **Reschedule Times**. Si vous supprimez le job, il ne sera pas reprogrammé. La valeur par défaut est **no**.

Cette spécification peut se révéler utile pour les pc portables ainsi que pour toutes les machines qui ne sont pas connectées au réseau en permanence.

Reschedule Interval = <time-specification> Si cette directive est activée, alors si le job se termine en erreur, il sera reprogrammé après l'intervalle de temps stipulé par **time-specification**. Consultez la section the time specification formats du chapitre

Configurer Bacula pour plus de détails sur les spécifications de temps. Si aucun intervalle n'est spécifié, le job ne sera pas reprogrammé en cas d'erreur.

Reschedule Times = <count> Cette directive précise le nombre maximal de tentatives d'exécution du job. S'il est fixé à zéro (valeur par défaut), le job sera reprogrammé indéfiniment.

Run = <job-name> La directive Run (à ne pas confondre avec l'option Run dans un Schedule) vous permet de démarrer ou de cloner des jobs. En utilisant les mots-clef de clonage (voir ci dessous), vous pouvez sauvegarder les mêmes données (ou presque les mêmes) vers deux (ou plus) lecteurs en même temps. Le nom de job **job-name** est en principe le même que celui de la ressource job courante (créant ainsi un clone). Cependant, ce peut être n'importe quel nom de job, de sorte qu'un job peut démarrer d'autre jobs liés. La partie après le signe égale doit être encadrée de double quotes, et peut contenir toute chaîne ou jeu d'options (surcharges) qui pourraient être spécifiées à l'utilisation de la commande Run dans la Console. Par exemple, **storage=DDS-4 ...**. De plus, deux mots-clef spéciaux vous permettent de cloner le job courant : **level=%l** et **since=%s**. le %l du mot clef level permet d'entrer le niveau réel du job courant et le %s du mot clef since permet d'imposer la même date pour la comparaison que celle utilisée par le job courant.a Notez que dans le cas du mot-clef since, le %s doit être encadré de double quotes, qui doivent être elles mêmes précédées de barres obliques arrières puisque elles sont déjà entre double quotes. Par exemple :

```
run = "Nightly-backup level=%s since=\"%s\" storage=DDS-4"
```

Un job cloné ne démarrera pas de nouveaux clones, aussi il n'est pas possible de les cascader.

Priority = <number> Cette directive vous permet de contrôler l'ordre d'exécution des jobs en spécifiant un entier positif non nul. Plus grand est ce nombre, plus basse est la priorité du job. En supposant que vous n'exécutiez pas de jobs simultanés, tous les jobs en file d'attente avec la priorité 1 seront exécutés avant ceux avec la priorité 2, et ainsi de suite, sans prise en compte de l'ordre original de planification.

La priorité affecte seulement les jobs en file d'attente, et non les jobs déja en cours d'exécution. Si un ou plusieurs jobs de priorité 2 sont déjà en cours d'exécution, et si un nouveau job est programmé avec la priorité 1, les jobs en cours d'exécution doivent se terminer pour que le job de priorité 1 puisse démarrer.

La priorité par défaut est 10.

Si vous voulez exécutez plusieurs jobs simultanés, ce qui n'est pas recommandé, vous devriez garder les points suivants à l'esprit :

- Pour exécuter plusieurs jobs simultanés, vous devez ajuster la directive **Maximum Concurrent Jobs** en cinq ou six endroits différents : dans le fichier bacula-dir.conf, les ressources **Job**, **Client**, **Storage**; dans le fichier bacula-fd, la ressource **FileDaemon**; et dans le fichier bacula-sd.conf, la ressource **Storage**. Si vous omettez l'un d'entre eux, les jobs seront exécutés un par un.
- Bacula n'exécute pas simultanément les jobs de priorités distinctes.
- Si Bacula exécute un job de priorité 2 et si un nouveau job de priorité 1 est programmé, il attendra la fin du job de priorité 1, même si les paramètres **Maximum Concurrent Jobs** pourraient permettre l'exécution simultanée de deux jobs.
- Supposons que Bacula soit en train d'exécuter un job de priorité 2 et qu'un job de priorité 1 soit programmé et mis en queue en attente de la fin du job de priorité 2. Si vous démarrez alors un second job de priorité 2, le job en attente de priorité 1 empèchera le nouveau job de priorité 2 de s'exécuter en parallèle au premier. Ainsi : tant qu'il reste un job de priorité supérieure à exécuter, aucun nouveau job de priorité inférieure ne pourra démarrer, même si les paramètres **Maximum Concurrent Jobs** devraient le permettre. Ceci permet d'assurer que les jobs de priorité supérieure seront exécutés dès que possible.

Si vous avez plusieurs jobs de priorités différentes, il est préférable de ne pas les démarrer exactement à la même heure, car Bacula doit les examiner un à la fois. Si, par hazard, Bacula commence par traiter un job de priorité inférieure, il sera exécuté avant votre job de priorité élevé. Pour éviter cette situation, démarrez l'un quelconque des jobs de priorité élevée quelques secondes avant ceux de basse priorité. Ainsi, vous serez assuré que Bacula examine les jobs dans l'ordre voulu et que votre schéma de priorités sera respecté.

Write Part After Job = <yes—no> Cette directive est implémentée depuis la version 1.37. Si la valeur de cette directive est **yes** (**no** par défaut), un nouveau "fichier partition" (ndt : part file) sera créé après la fin du job.

Cette directive devrait être activée lors de l'écriture sur des périphérique qui requièrent un montage (par exemple, les DVDs),

afin de vous assurer que le fichier partition courant, celui qui contient les données de ce job, est envoyé vers le périphérique, et qu'aucune donnée n'est laissée dans le fichier temporaire sur le disque dur. Quoi qu'il en soit, avec certains supports tels que les DVD+R et DVD-R, beaucoup d'espace (environ 10 Mb) est perdu à chaque fois qu'un fichier partition est écrit. Aussi, si vous exécutez plusieurs jobs à la suite, vous devriez régler cette directive à **no** pour tous ces jobs sauf le dernier, pour éviter un gaspillage important d'espace, tout en ayant la certitude que les données sont bien écrites sur le médium lorsque tous les jobs sont achevés.

Cette directive est ignorée avec les bandes et les périphériques FIFO.

Voici un exemple de définition de ressource Job valide.

```
Job {
  Name = "Minou"
  Type = Backup
  Level = Incremental                 # default
  Client = Minou
  FileSet="Minou Full Set"
  Storage = DLTDrive
  Pool = Default
  Schedule = "MinouWeeklyCycle"
  Messages = Standard
}
```

La ressource JobDefs

La ressource Jobdefs admet toutes les directives qui peuvent apparaître dans une ressource Job. Une ressource Jobdefs ne créé en aucun cas un Job, son rôle est de pouvoir être désignée dans une ressource Job comme un ensemble de paramètres par défaut. Ceci permet de définir plusieurs jobs similaires avec concision, en ne mentionnant, pour chaque job, que les différences avec les valeurs par défaut spécifiées dans la ressource Jobdefs.

La ressource Schedule

La ressource Schedule offre un moyen pour planifier automatiquement un Job, mais aussi la possibilité de surcharger les paramètres par défaut de Level, Pool, Storage, et Messages ressources. Si une ressource Schedule n'est pas spécifiée dans un job, ce job ne peut être exécuté que manuellement. En général, vous spécifierez une action et le moment de son lancement.

Schedule Début des directives Schedule. La ressource **Schedule** n'est pas requise, mais il vous en faudra au moins une si vous souhaitez que vos jobs soient exécutés automatiquement.

Name = <name> Le nom du Schedule défini. Cette directive est requise.

Run = <Job-overrides> <Date-time-specification> La directive Run définit quand un job doit être exécuté, et les éventuelles surcharges à appliquer. Il est possible de spécifier plusieurs directives **run** au sein d'une ressource **Schedule**, elles seront toutes appliquées. Si vous avez deux directives **Run** qui démarrent au même moment, deux jobs seront lancés simultanément (en fait, avec une seconde d'écart).

La directive **Job-overrides** permet d'outrepasser les spécifications de Level, Storage, Messages et Pool écrites dans la ressource Job. De plus, les spécifications FullPool, DifferentialPool et IncrementalPool permettent de passer outre les spécification de Pool, en accord avec le niveau (level) effectif d'exécution du job.

L'utilisation de surcharges permet de peaufiner le paramétrage d'un job particulier. Par exemple, vous pourriez surcharger une spécification Messages qui enverrait vos logs de backups vers un fichier, de faon à ce qu'ils vous soient envoyés par mails pour les Fulls hebdomadaires ou mensuelles.

Les directives **Job-overrides** sont spécifiées en tant que **mot-clef=valeur** où le mot-clef est l'un des suivants : Level, Storage, Messages, Pool, FullPool, DifferentialPool ou IncrementalPool, et la **valeur** est définie selon le format adapté à la directive. Vous pouvez spécifier plusieurs surcharges **Job-overrides** en une seule directive **Run** en les séparant par des espaces ou des trailing comas (traduction ?). Par exemple :

Level=Full Tous les fichiers du FileSet qu'ils aient ou non changé.

Level=Incremental Tous les fichiers qui ont changé depuis la dernière sauvegarde.

Pool=Weekly Spécifie l'utilisation du Pool nommé **Weekly**.

Storage=DLT_Drive Spécifie l'utilisation du lecteur **DLT_Drive** pour périphérique de stockage.

Messages=Verbose Spécifie l'utilisation de la ressource messages **Verbose** pour le job.

FullPool=Full Spécifie l'utilisation du Pool nommé **Full** si le job est une sauvegarde Full, ou s'il a été élevé en Full bien qu'ayant été lancé en tant que différentiel ou incrémental.

DifferentialPool=Differential Spécifie l'utilisation du Pool nommé **Differential** si le job est une sauvegarde différentielle.

IncrementalPool=Incremental Spécifie l'utilisation du Pool nommé **Incremental** si le job est une sauvegarde incrémentale.

SpoolData=yes—no Indique à Bacula d'ordonner au Storage Daemon de placer les données sur un spool disque avant de les envoyer vers les cartouches.

WritePartAfterJob=yes—no Indique à Bacula d'ordonner au Storage Daemon d'écrire le fichier partition courant vers le périphérique lorsque le job s'achève (voir la directive Write Part After Job dans la ressource Job).

Date-time-specification Détermine la planification d'exécution du job. La spécification est une répétition, et, par défaut, Bacula est paramétré pour exécuter un job au début de chaque heure de chaque jour de chaque semaine de chaque mois de chaque année. Ce n'est probablement pas ce que vous souhaitez, aussi vous devez préciser ou limiter les moments où vous souhaitez voir vos jobs exécutés. Toute spécification est supposée cyclique et servira à limiter le cycle par défaut. Ceci se fait en spécifiant des masques ou des horaires, jours de la semaine, jours du mois, semaines du mois, semaines de l'année et mois de l'année où vous voulez exécuter le job. En combinant ces possibilités, vous pouvez définir une planification qui se répète à presque n'importe quelle fréquence.

Concrètement, vous devez définir les **mois**, **jour**, **heure** et **minute** où le job est à exécuter. Parmis ces quatre objets, le **jour** est particulier en ce qu'il peut spécifier un jour du mois (1,2,...31) ou de la semaine (Monday, Tuesday,...Sunday). Enfin, vous pouvez aussi spécifier un jour de la semaine pour restreindre la planification à la première, deuxième, troisième, quatrième ou cinquième semaine du mois.

Par exemple, si vous spécifiez seulement un jour de la semaine, disons **Mardi**, le job sera exécuté toutes les heures de chaque mardi de chaque mois. La raison en est que les paramètres **Mois** et **Heure** sont restés à leurs valeurs par défaut : chaque mois et chaque heure.

Notez que, par défaut, sans autre spécification, votre job s'exécutera au début de chaque heure. Si vous souhaitez que votre job s'exécute plus souvent qu'une fois par heure, il vous faudra définir plusieurs spécifications **run** avec pour chacune une minut différente.

Les dates et horaires d'exécutions des jobs peuvent être spécifiés comme suit, en pseudo-BNF :

```
<void-keyword>    = on
```

```
<at-keyword>      = at
<week-keyword>    = 1st | 2nd | 3rd | 4th | 5th | first |
                    second | third | forth | fifth
<wday-keyword>    = sun | mon | tue | wed | thu | fri | sat |
                    sunday | monday | tuesday | wednesday |
                    thursday | friday | saturday
<week-of-year-keyword> = w00 | w01 | ... w52 | w53
<month-keyword>   = jan | feb | mar | apr | may | jun | jul |
                    aug | sep | oct | nov | dec | january |
                    february | ... | december
<daily-keyword>   = daily
<weekly-keyword>  = weekly
<monthly-keyword> = monthly
<hourly-keyword>  = hourly
<digit>           = 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
<number>          = <digit> | <digit><number>
<12hour>          = 0 | 1 | 2 | ... 12
<hour>            = 0 | 1 | 2 | ... 23
<minute>          = 0 | 1 | 2 | ... 59
<day>             = 1 | 2 | ... 31
<time>            = <hour>:<minute> |
                    <12hour>:<minute>am |
                    <12hour>:<minute>pm
<time-spec>       = <at-keyword> <time> |
                    <hourly-keyword>
<date-keyword>    = <void-keyword>  <weekly-keyword>
<day-range>       = <day>-<day>
<month-range>     = <month-keyword>-<month-keyword>
<wday-range>      = <wday-keyword>-<wday-keyword>
<range>           = <day-range> | <month-range> |
                          <wday-range>
<date>            = <date-keyword> | <day> | <range>
<date-spec>       = <date> | <date-spec>
<day-spec>        = <day> | <wday-keyword> |
                    <day-range> | <wday-range> |
                    <week-keyword> <wday-keyword>
                    <day-range> | <wday-range> |
                    <daily-keyword>
<month-spec>      = <month-keyword> | <month-range> |
                    <monthly-keyword>
<date-time-spec>  = <month-spec> <day-spec> <time-spec>
```

Notez que les spécifications de semaine et d'année suivent les définitions ISO standard de semaine et année, où la semaine 1 est la semaine qui contient le premier jeudi de l'année, ou alternativement, la semaine qui contient le quatrième jour de janvier. Les semaines sont numérotées w01 à w53. w00 est pour Bacula la semaine qui précède la première semaine ISO (c'est à dire celle qui contient les quelques premiers jours de l'année si aucun n'est un jeudi). w00 n'est pas définie dans les spécifications ISO. Une semaine commence le Lundi et se termine le Dimanche.
Voici un exemple de ressource Schedule nommée **WeeklyCycle** qui exécute un job de niveau Full chaque Dimanche à 1h05 et un job de niveau

incrémental du Lundi au Samedi à 1h05 :

```
Schedule {
  Name = "WeeklyCycle"
  Run = Level=Full sun at 1:05
  Run = Level=Incremental mon-sat at 1:05
}
```

Voici un exemple de cycle mensuel :

```
Schedule {
  Name = "MonthlyCycle"
  Run = Level=Full Pool=Monthly 1st sun at 1:05
  Run = Level=Differential 2nd-5th sun at 1:05
  Run = Level=Incremental Pool=Daily mon-sat at 1:05
}
```

Le premier de chaque mois :

```
Schedule {
  Name = "First"
  Run = Level=Full on 1 at 1:05
  Run = Level=Incremental on 2-31 at 1:05
}
```

Toutes les dix minutes :

```
Schedule {
  Name = "TenMinutes"
  Run = Level=Full hourly at 0:05
  Run = Level=Full hourly at 0:15
  Run = Level=Full hourly at 0:25
  Run = Level=Full hourly at 0:35
  Run = Level=Full hourly at 0:45
  Run = Level=Full hourly at 0:55
}
```

Notes techniques sur les Schedules

Au niveau interne, Bacula considère un schedule en tant que bit masque. Il y a six masques et un champ minute pour chaque schedule. Les masques sont heure, jour du mois (mday), jour de la semaine (wday), semaine du mois (wom), et semaine de l'année (woy). Le schedule est initialisé de faon à avoir les bits de chacun de ces masques positionnés, ce qui signifie qu'au début de chaque heure, le job sera exécuté. Quand vous spécifiez un mois pour la première fois, le masque est effacé et le bit correspondant au mois sélectionné est ajouté au masque. Si vous spécifiez un second mois, le bit correspondant est aussi ajouté. Ainsi, lorsque Bacula examine le masque pour voir si les bits placés correspondent à la date courante, votre job ne sera exécuté que pendant les deux mois que vous avez spécifiés. De même, si vous spécifiez un horaire, le masque Heure est effacé et le bit correspondant

à l'heure que vous avez spécifiée est placé, les minutes sont quant à elles stockées dans le champ Minutes.
Pour chacun de vos schedules, vous pouvez visualiser le masque associé grce à la commande **show schedules** du programme Console. Notez que le bit masque est "zero based", et que Dimanche est le premier jour de la semaine (bit 0)

La ressource FileSet

La ressource FileSet définit les fichiers à inclure dans une sauvegarde. Pour chaque job de type sauvegarde, il est nécessaire de définir au moins une ressource **FileSet**. Un **FileSet** consiste en une liste de fichiers ou répertoires à inclure, une liste de fichiers ou répertoires à exclure, et diverses options de sauvegardes telles que compression, chiffrement et signatures qui doivent être appliquées à chaque fichier.
Toute modification de la liste des fichiers inclus provoque la création par Bacula d'un nouveau FileSet (défini par le nom et la somme de contrôle MD5 du contenu du paragraphe Include). Chaque fois qu'un nouveau FileSet est créé, Bacula s'assure que la première sauvegarde est une Full.

FileSet Début de la ressource FileSet. Au moins une ressource **FileSet** doit être définie.

Name = <name> Le nom de la ressource FileSet. Cette directive est requise.

Ignore FileSet Changes = <yes—no> Si cette directive est activée (**yes**), toute modification des listes d'inclusion ou d'exclusion du FileSet sera ignorée et Bacula n'élèvera pas la prochaine sauvegarde en Full. La valeur par défaut est **no**, ainsi, si vous modifiez une des listes d'inclusion ou d'exclusion du FileSet, Bacula forcera une sauvegarde Full pour assurer que tout soit bien sauvegardé proprement. Il n'est pas recommandé d'activer cette directive. Cette directive est disponible à partir de Bacula 1.35.4.

Include { [Options {<file-options>} ...] <file-list> }

Options { <file-options> }

Exclude { <file-list> }

La ressource Include doit contenir une liste de répertoires et/ou fichiers à traiter lors de la sauvegarde. Normalement, tous les fichiers trouvés dans tous les sous-répertoires de tout répertoire de la liste d'inclusion des fichiers seront sauvegardés. La ressource Include peut aussi comporter une ou plusieurs ressources Options qui spécifient des paramètres tels que la compression à appliquer à tous les fichiers ou à n'importe quel sous ensemble de fichiers à sauvegarder.

Le nombre de ressources **Include** par FileSet n'est pas limité, chacune ayant sa propre liste de répertoires et/ou fichiers à sauvegarder et ses propres paramètres définis par une ou plusieurs ressources Options. La liste de fichiers **file-list** consiste en un nom de fichier ou répertoire par ligne. Les noms de répertoire doivent être spécifiés sans slash final.
Vous devez toujours spécifier des chemins absolus pour tout fichier ou répertoire que vous placez dans un FileSet. De plus, sur les machines Windows, vous devez **toujours** préfixer le répertoire ou nom de fichier d'une spécification de disque (par exemple : **c:/xxx**) en utilisant le séparateur de répertoire Unix (slash /).
Le comportement par défaut de Bacula en ce qui concerne le traitement des répertoires est de descendre récursivement dans chaque répertoire et de sauvegarder tous les fichiers et sous-répertoires. Par défaut, Bacula ne suit pas les systèmes de fichiers transverses (en terminologie Unix, les points de montage). Ceci signifie que si vous spécifiez la partition racine (par exemple /), Bacula sauvegardera seulement la partition racine, et aucun des systèmes de fichiers montés. De faon analogue, sur les systèmes Windows, vous devez expliciter chacun des disques que vous souhaitez sauvegarder (par exemple **c:/** et **d:/**...). De plus, au moins pour les systèmes Windows, il sera la plupart du temps nécessaire d'encadrer chaque spécification de doubles quotes, particulièrement si le nom du répertoire (ou du fichier) comporte des espaces. La commande **df** des systèmes Unix vous fournira la liste des répertoires qu'il vous faudra spécifier pour tout sauvegarder. Voyez ci-dessous pour un exemple.
Soyez attentif à ne pas inclure un répertoire deux fois, car il serait sauvegardé deux fois, ce qui gaspillerait l'espace sur votre périphérique de sauvegarde. Cette erreur est facile à commettre. Par exemple :

```
Include {
  File = /
  File = /usr
  Options { compression=GZIP }
}
```

Sur un système Unix où /usr est un sous répertoire (plutôt qu'un système de fichiers monté), cette ressource Include sauvegarderait /usr deux fois. Dans ce cas, sur les versions antérieures à 1.32f-5-09Mar04, en raison d'un bug, vous ne pourriez restaurer les fichiers liés physiquement sauvegardés deux fois.
Si vous avez utilisé des versions de Bacula antérieures à 1.34.3, vous noterez ces modifications dans la syntaxe des FileSets :

1. il n'y a pas de signe égale (=) après le "include" et avant l'accolade ouvrante ({) ;
2. chaque répertoire (ou nom de fichier) à sauvegarder est précédé de "**File =**" ;

3. les options qui apparaissaient précédemmant sur la ligne Include doivent désormais être spécifiées dans leur propre ressource Options.

La ressource Options est optionnelle, mais lorsqu'elle est spécifiée, elle doit contenir une liste de lignes "**mot-clef=valeur**" relatives aux options à appliquer à la liste de fichiers/répertoires. Plusieurs ressources Options peuvent être spécifiées l'une après l'autre. Lorsqu'un fichier se trouve dans un dossier spécifié, les options sont appliquées au nom de fichier pour savoir s'il doit être sauvegardé, et comment. Les ressources Options sont appliquées dans l'ordre où elles apparaîssent dans le FileSet jusqu'à ce qu'il y en ait une qui corresponde. Une ressource Options qui ne contient pas de directive **wild** (spécification de caractère joker, voir ci-dessous) est considérée comme concernant tous les fichiers. Il est important de bien comprendre ceci, car une fois que Bacula a déterminé que des Options s'appliquent à un fichier donné, ce fichier sera sauvegardé sans tenir compte d'aucunes des éventuelles autres ressources Options. Ceci signifie que toute ressource Options avec caractères joker doit apparaître avant une ressource Options sans caractères joker.
Si, pour quelque raison, Bacula applique toutes les ressources Options à un fichier sans qu'aucune ne corresponde (en général à cause de caractères joker qui ne correspondent pas), par défaut Bacula sauvegardera le fichier. Ceci est assez logique si vous considérez la situation sans options, où vous souhaitez que tout soit sauvegardé. De plus, dans le cas ou aucune correspondance n'est trouvée, Bacula utilise les options de la dernière ressource Options. Par conséquent, si vous souhaitez définir un jeu d'options par défaut, vous devriez les placer dans la dernière ressource Options.
Les directives disponibles pour les ressources Options sont les suivantes :

compression=GZIP Tous les fichiers sauvegardés sont compressés (NDT : compression logicielle, par opposition à la compression matérielle effectuée par le lecteur) au format GNU ZIP. Chaque fichier est compressé individuellement par le File Daemon. S'il y a un problème à la lecture d'une cartouche au niveau de l'enregistrement d'un fichier, il affectera tout au plus ce fichier et aucun des autres fichiers de la cartouche. La plupart du temps, cette option n'est pas nécessaire si vous avez un lecteur de bandes moderne qui applique sa propre compression. En fait, si vous activez les deux compressions simultanément, il se peut que vos fichiers occupent plus d'espace sur le volume qu'avec une seule.

La compression logicielle est particulièrement intéressante lorsque vous sauvegardez sur disque, et peut être d'un grand secours si vous avez un ordinateur rapide mais un réseau lent.

La spécification **GZIP** utilise le niveau de compression six par

défaut (i.e. **GZIP** est équivalent à **GZIP6**). Si vous voulez utiliser un niveau différent (de 1 à 9), vous pouvez le spécifier en ajoutant le numéro du niveau voulu à la fin du mot **GZIP**, sans espace. Ainsi, **compression=GZIP1** désigne la compression la moins efficace, mais l'algorithme le plus rapide, tandis que **compression=GZIP9** est le niveau de compression le plus élevé, mais requière plus de puissance de calcul. Selon la documentation GZIP, les niveaux de compression supérieurs à 6 ne procurent généralement que peu de compression supplémentaire alors qu'ils sont plutôt exigeants en puissance de calcul.

signature=SHA1 La signature SHA1 est calculée pour tous les fichiers sauvegardés. L'algorithme SHA1 est réputé plus lent que MD5, mais bien meilleur d'un point de vue cryptographique (i.e. beaucoup moins de collisions et probabilité de piratage bien inférieure.). Nous recommandons fortement d'activer l'une ou l'autre des options SHA1 ou MD5 par défaut pour tous les fichiers. Notez que seule l'une de ces deux options peut être activée pour tout fichier.

signature=MD5 La signature MD5 est calculée pour tous les fichiers sauvegardés. Activer cette option résulte en une charge CPU supplémentaire de l'ordre de 5% pour chaque fichier sauvegardé. D'autre part, la signature MD5 ajoute 16 octets supplémentaires au catalogue pour chaque fichier sauvegardé. Nous recommandons fortement d'activer l'une ou l'autre des options SHA1 ou MD5 par défaut pour tous les fichiers.

verify=<options> Les "options-lettres" sont utilisées lors de l'exécution de jobs de type **Verify** de niveau **Level=Catalog** et de niveau **Level=DiskToCatalog**. Les options peuvent être n'importe quelle combinaison de ces lettres.

i compare les inodes

p compare bits de permissions

n compare le nombre de liens

u compare les user ids

g compare les group ids

s compare les tailles

a compare les date d'accès (access time)

m compare les dates de modification (st_mtime)

c compare les dates de changement (st_ctime)

s signale tout fichier dont la taille a diminué

5 compare les signatures MD5

1 compare les signatures SHA1

Le jeu d'options **pins5** (qui compare les bits de permissions, les inodes, les nombres de liens, la taille des fichiers et les signatures MD5) est très utile pour des jobs de type verify de niveaux **Level=Catalog** ou **Level=DiskToCatalog**.

onefs=yes—no Si cette option est activée (valeur **yes**, par défaut), Bacula ne changera pas de système de fichiers. Autrement dit, il ne sauvegardera pas les systèmes de fichiers montés sur des sous-répertoires. Si vous souhaitez sauvegarder plusieurs systèmes de fichiers, vous pouvez les énumérer explicitement. Une autre possibilité consiste à désactiver l'option onefs (**onefs=no**) afin que Bacula sauvegarde les systèmes de fichiers montés trouvés dans les répertoires listés dans votre FileSet. Ainsi, si vous avez des systèmes de fichiers NFS ou Samba montés sur un répertoire listé dans le FileSet, ils seront aussi sauvegardés. En principe, il est préférable d'activer cette option et de nommer explicitement chaque système de fichier que vous voulez sauvegarder. Ce nommage explicite évite le risque de tomber dans une boucle infinie de systèmes de fichiers. Voyez l'exemple ci-dessous pour plus de détails.

portable=yes—no Si cette option est activée (la valeur par défaut est **no**), le File Daemon sauvegarde les fichiers win32 dans un format portable, mais tous les attributs de fichiers win32 ne seront pas sauvegardés ni restaurables. La valeur par défaut est **no**, ce qui signifie que sur les systèmes Win32, les données sont sauvegardées en utilisant les appels Windows API et sur les WinNT/2k/XP, tous les attributs de sécurité et de propriété sont correctement sauvegardés et restaurés. Cependant, ce format n'est pas portable aux autres systèmes – par exemple UNIX, Win95/98/Me. Lors de la sauvegarde de systèmes Unix, cette option est ignorée, et à moins que vous n'ayez un besoin spécifique de portabilité de vos sauvegardes, nous recommandons d'accepter la valeur par défaut (**no**) de sorte qu'un maximum d'informations concernant vos fichiers soit sauvegardé.

recurse=yes—no Si cette option est activée (la valeur par défaut est **yes**), Bacula descend récursivement dans tout sous-répertoire trouvé, à moins qu'il ne soit explicitement exclu par une définition **exclude**. Si vous désactivez cette option (**recurse=no**), Bacula sauvegardera toutes les entrées de sous- répertoires, mais n'entrera pas dans ces sous-répertoires, et ainsi ne sauvegardera pas les fichiers ou épertoires contenus dans ces sous-répertoires. En principe, vous préfèrerez la valeur par défaut (**yes**).

sparse=yes—no Cette option active un code spécial qui détecte les fichiers clairsemés tels ceux créés par ndbm. Elle est désactivée

par défaut (**sparse=no**), de sorte qu'aucun contrôle n'est fait pour rechercher les fichiers clairsemés. Vous pouvez l'activer sans danger sur des fichiers non clairsemés, cependant elle entraîne une légère charge supplémentaire pour la détection de tampons remplis de zéros (buffers of all zero), et un léger surplus d'espace sur l'archive de sortie sera utilisé pour ssauver les adresses de recherche de chaque enregistrement non-nul trouvé.

Restrictions: Bacula lit les fichiers dans des tampons de 32K. Si le tampon entier est rempli de zéros, il sera traité en tant que bloc clairsemé, et ne sera pas écrit sur la cartouche. En revanche, si une partie quelconque du tampon est non-nulle, le tampon sera intégralement copié sur la cartouche, avec éventuellement des secteurs de disque (généralement 4098 octets) entièrement nuls. La détection par Bacula des blocs clairsemés a lieu sur des blocs de 32K plutôt que sur des blocs de taille déterminée par le système. Si quelqu'un considère ceci comme un réelle problème, merci d'envoyer une demande de modification en exposant les raisons. Ce code est apparu avec la version 1.27 de Bacula.

Si vous n'êtes pas familier avec les notions de fichiers clairsemés, prenons pour exemple un fichier où vous écrivez 512 octets à l'adresse 0, puis 512 octets à l'adresse 1 million. Le système d'exploitation n'allouera que deux blocs, et rien n'est alloué pour l'espace vide. Pourtant, lorsque vous lisez le fichier clairsemé, le système retourne tous les zéros comme si l'espace était alloué, et si vous sauvegardez un tel fichier, vous utiliserez beaucoup d'espace sur le volume pour écrire des zéros. Pire encore, lorsque vous restaurez ce fichier à son emplacement initial, tous les emplacements précédemment vides seront cette fois alloués, occupant ainsi beaucoup plus d'espace disque. En activant l'option **sparse**, Bacula recherchera spécifiquement l'espace vide dans les fichiers afin d'éviter ces inconvénients. Le prix à payer est que Bacula doit d'abord examiner chaque bloc lu avant de l'écrire. Sur un système lent, ceci peut-être important. Si vous suspectez certains de vos fichiers d'être clairsemés, vous devriez mesurer les performances et gains d'espace avec et sans l'options, ou ne l'activer que pour les fichiers effectivement clairsemés.

readfifo=yes—no Cette option, si elle est activée, indique au client de lire les données (lors d'une sauvegarde) et de les écrire (lors d'une restauration) sur un FIFO (pipe) explicitement explicitement mentionné dans le FileSet. Dans ce cas, vous devez avoir un programme actif qui écrit sur ce FIFO dans le cas d'une sauvegarde, ou qui le lit dans le cas d'une restauration. (Ceci peut être accompli par la directive **RunBeforeJob**). Si

cette condition n'est pas satisfaite, Bacula demeurera en suspens indéfiniment en lecture/écriture du FIFO. Lorsque cette option est désactivée (par défaut), le Client sauvegarde simplement l'entrée du répertoire pour le FIFO.

mtimeonly=yes—**no** Cette option, si elle est activée, indique au client que la sélection de fichiers lors d'une sauvegarde incrémentale ou différentielle ne doit se référer qu'aux valeurs de st_mtime du paquet stat(). La valeur par défaut est **no**, ce qui signifie que la sélection de fichiers à sauvegarder se base sur les deux valeurs st_mtime et st_ctime. En général, il n'est pas recommandé d'activer cette option.

keepatime=yes—**no** Avec cette option activée, Bacula rétablit le champ st_atime (date d'accès) des fichiers qu'il sauvegarde à leur valeur d'avant la sauvegarde. Cette option n'est généralement pas recommandée car il existe peu de programmes qui utilisent st_atime, et la charge de la sauvegarde se trouve augmentée par les appels systèmes nécessaires pour rétablir les dates. (Je ne suis pas sur que ceci fonctionne sous Win32).

wild=<string> Spécifie une chaîne de caractères jokers à appliquer aux fichiers. Notez que si **Exclude** n'est pas activée, cette chaîne sélectionnera les fichiers à sauvegarder. Si au contraire **Exclude=yes** est spécifié, la chaîne sélectionnera les fichiers à exclure de la sauvegarde. Plusieurs directives wild-card peuvent être spécifiées et sont appliquées séquentiellement jusqu'à ce que l'une d'elles corresponde.

regex=<string> Spécifie une expression régulière étendue POSIX à appliquer aux fichiers. Cette directive est disponible à partir de Bacula 1.35. Si **Exclude** n'est pas activée, cette expression régulière sélectionnera les fichiers à sauvegarder. Si au contraire **Exclude=yes** est spécifié, elle sélectionnera les fichiers à exclure de la sauvegarde. Plusieurs directives regex peuvent être spécifiées et sont appliquées séquentiellement jusqu'à ce que l'une d'elles corresponde.

exclude=yes—**no** Lorsque cette option est activée, tout fichier qui correspond aux options est exclu de la sauvegarde. La valeur par défaut est **no**.

aclsupport=yes—**no** Si cette option est activée, et si vous avez installé la librarie POSIX **libacl** sur votre système, Bacula sauvegardera Listes de Contrôles d'Accès (ACL) UNIX des fichiers et répertoires telles que définies dans IEEE Std 1003.1e version 17 et "POSIX.1e" (abandonné). Cette fonction n'est disponible que sur UNIX et dépend de la librairie ACL. Bacula est automatiquement compilé avec le support ACL si la librairie **libacl** est installée sur

votre système (ceci est reporté dans le fichier config.out). Lors de la restauration, Bacula tentera de restaurer les ACLs. S'il n'y a pas de support ACL sur le système cible, Bacula ne restaurera que les fichiers et répertoires sans les informations ACL. Veuillez noter que si vous sauvegardez un système de fichiers EXT3 ou XFS avec le support des ACLs, et que vous restaurez vers un système de fichiers sans ACLs (tel , peut-être reiserfs), les ACLs seront ignorées.

<**file-list**> est une liste de répertoires et/ou noms de fichiers spécifiés avec la directive **File =**. Pour inclure des noms contenant des espaces, entourez-les de guillemets (doubles quotes).
Il existe quelques notations particulières pour spécifier des fichiers et répertoires dans une liste de fichiers **file-list**. Les voici :

- Tout nom précédé d'un signe "at" (@) est compris comme le nom d'un fichier, lequel contient une liste de fichiers, chacun précédé d'une directive "File=". Ce fichier est lu lorsque le fichier de configuration est parcouru au démarrage du Director. Notez bien que le fichier est lu sur sur la machine qui héberge le Director (autrement dit, le serveur de sauvegardes) et non sur le Client. En fait, le "@NomDeFichier" peut apparaître n'importe où dans le fichier de configuration où un objet pourrait être lu, le contenu du fichier désigné sera logiquement inséré à l'emplacement du "@NomDeFichier". Ce qui doit figurer dans le fichier dépend de l'emplacement du "@NomDeFichier" au sein du fichier de configuration.
- Tout nom précédé d'une barre verticale (—) est compris comme le nom d'un programme. Ce programme sera exécuté sur la machine qui héberge le Director au moment où le job démarre (et non lorsque le Director lit son fichier de configuration), et toute sortie de ce programme sera peru en tant que liste de fichiers ou répertoires, un par ligne, à inclure. Ceci vous permet d'avoir un job qui, par exemple, inclue toutes les partitions locales même si vous changez le partitionnement en ajoutant des disques. En général, il vous faudra précéder votre commande d'un "**sh -c**" afin qu'elle soit invoquée par un shell. Ce ne sera pas le cas si vous invoquez un script comme dans le second exemple ci-dessous. Vous devez aussi prendre soin d'échapper (précéder d'un \) les caractères jokers, les caractères du shell, ainsi que toute espace dans votre commande. Si vous utilisez des simples quotes (') dans des doubles quotes (“), Bacula traitera tout ce qui est entre simples quotes comme un seul champ, et il ne dera donc pas nécessaire d'échapper les espaces. En général, parvenir à avoir toutes les quotes et échappements corrects est un calvaire, comme vous pouvez le constater dans le prochain exemple. Par conséquent, il est

souvent plus facile de tout mettre dans un fichier et d'utiliser simplement le nom de fichier dans Bacula. Dans ce cas le "**sh -c**" ne sera plus nécessaire, pourvu que la première ligne du fichier soit **#!/bin/sh**.
Par exemple :

```
Include {
   Options { signature = SHA1 }
   File = "|sh -c 'df -l | grep \"^/dev/hd[ab]\" | grep -v \".*/tmp\" \
      | awk \"{print \\$6}\"'"
}
```

produira une liste de toutes les partitions locales sur un système RedHat Linux. Notez que la ligne si dessus a été coupée, mais devrait normalement être écrite sur une seule ligne. Quoter est un réel problème car vous devez d'une part le faire pour Bacula - ce qui consiste à précéder tout \ et tout " avec un \ - et d'autre part pour les commandes shell. En définitive, il est probablement plus aisé d'exécuter un petit fichier tel que :

```
Include {
  Options {
    signature=MD5
  }
  File = "|my_partitions"
}
```

où le fichier my_partitions contient :

```
#!/bin/sh
df -l | grep "^/dev/hd[ab]" | grep -v ".*/tmp" \
      | awk "{print \$6}"
```

Si la barre verticale (—) devant "my_partitions" est précédée d'une barre oblique (\), le programme sera exécuté sur la machine cliente plutôt que sur la machine hébergeant le Director – (ceci est implémenté, mais n'est pas complètement testé, et a été rapporté fonctionner sous Windows). Veuillez noter que si le nom de fichier est donné entre quotes, vous devrez utiliser deux barres obliques. Voci un exemple, fourni par John Donagher, qui sauvegarde toutes les partitions UFS locales sur un système distant :

```
FileSet {
  Name = "All local partitions"
  Include {
    Options { signature=SHA1; onefs=yes; }
    File = "\\|bash -c \"df -klF ufs | tail +2 | awk '{print \$6}'\""
  }
}
```

Notez que deux barres obliques \ sont requises après les doubles quotes (l'une préserve l'autre). Si vous utilisez Linux, changez simplement **ufs** en **ext3** (ou votre système de fichiers préféré) et l'affaire sera dans le sac.

– Tout élément de la liste de fichiers file-list précédé par un signe "inférieur" (<) est interprété comme un fichier qui sera lu sur la machine qui héberge le Director au moment où le job démarre. Son contenu est supposé être une liste de répertoires ou fichiers, un par ligne, à inclure dans la sauvegarde. Les noms ne doivent pas être quotés, même s'ils comportent des espaces. Cette fonction vous permet de modifier le fichier externe, et ainsi ce qui est sauvegardé sans avoir à redémarrer Bacula comme il le faudrait avec le modificateur @ décrit plus haut.
  Si vous précédez le signe "inférieur" (<) d'une barre oblique \<, le fichier mentionné sera lu sur la machine cliente au lieu de celle hébergeant le Director. Veullez noter que si le nom de fichier est donné entre quotes, il vous faudra utiliser deux barres obliques.
– Si vous spécifiez explicitement un block device (NDT ?) tel que **/dev/hda1**, alors Bacula considèrera ceci comme une partition raw à sauvegarder. Dans ce cas, vous êtes fortement invité à utiliser l'option **sparse=yes**, faute de quoi vous sauvegarderez la partition entière plutôt que seulement les données réellement contenues dans la partition. Par exemple :

```
Include {
  Options { signature=MD5; sparse=yes }
  File = /dev/hd6
}
```

  va sauvegarder les données de la partition /dev/hd6.
  will backup the data in device /dev/hd6.
  Ludovic Strappazon a fait remarquer que cette fonction pouvait servir à sauvegarder un disque Microsoft Windows. Il suffit de booter avec un Linux Rescue Disk, puis de charger un client Bacula statiquement lié comme décrit dans le chapitre Disaster Recovery avec Bacula de ce manuel. Sauvegardez alors la partition complète. En cas de désastre, vous pouvez alors restaurer la partition désirée en bootant une fois encore sur le Linux Rescue Disk et en utilisant le client Bacula statiquement lié.
– Si vous spécifiez explicitement un périphérique FIFO nommé (créé avec mkfifo), ets si vous ajoutez l'option **readfifo=yes**, Bacula lira le FIFO et sauvegardera ses données sur le volume. Par exemple :

```
Include {
  Options {
```

```
    signature=SHA1
    readfifo=yes
  }
  File = /home/abc/fifo
}
```

si **/home/abc/fifo** est un FIFO, Bacula va l'ouvrir, le lire, et stocker toutes les données ainsi obtenues sur le volume. Notez qu'il faut que vous ayez un processus qui écrit sur le FIFO, faute de quoi Bacula restera en suspens, et abandonnera au bout d'une minute pour passer au fichier suivant. Les données lues peuvent être de nature quelconque puisque Bacula les traite comme un flux.

Cette fonction est un excellent moyen de faire une sauvegarde "à chaud" d'une très grosse base de données. Vous pouvez utiliser la directive **RunBeforeJob** pour créer le FIFO et démarrer un programme qui lit dynamiquement votre base de données et l'écrit sur le FIFO. Bacula l'écrira alors sur le volume.

Lors de l'opération de restauration, l'inverse se produit : après que Bacula ait créé le FIFO, s'il y avait des données stockées par son biais (inutile de les lister explicitement ni d'ajouter aucune option), elles seront renvoyées vers le FIFO. Par conséquent, s'il existe un tel FIFO à restaurer, vous devez vous assurer qu'il y a un programme lecteur ou Bacula se bloquera et passera au fichier suivant après une minute.

Voici un exemple de définition de ressource FileSet valide. Notez que le premier Include insère le contenu du fichier **/etc/backup.list** lors du démarrage de Bacula (i.e. le @).

```
FileSet {
  Name = "Full Set"
  Include {
    Options {
      Compression=GZIP
      signature=SHA1
      Sparse = yes
    }
    File = @/etc/backup.list
  }
  Include {
     Options {
        wild = *.o
        Exclude = yes
     }
     File = /root/myfile
     File = /usr/lib/another_file
  }
```

```
}
```

Notez que dans l'exemple ci-dessus, tous les fichiers mentionnés dans /etc/backup.list seront compressé avec GZIP, qu'une signature SHA1 sera calculée sur le contenu des fichiers (leurs données), et que la prise en charge particulière des fichiers clairsemés (sparse) s'appliquera.

Les deux répertoires /root/myfile et /usr/lib/another_file seront aussi sauvegardés sans aucune option, mais tous les fichiers à extension **.o** de ces répertoires seront exlus de la sauvegarde.

Supposons que vous vouliez sauvegarder tout sauf **/tmp** sur votre système. La commande **df** vous fournit le résultat suivant :

```
[kern@rufus k]$ df
Filesystem      1k-blocks      Used Available Use% Mounted on
/dev/hda5         5044156    439232   4348692  10% /
/dev/hda1           62193      4935     54047   9% /boot
/dev/hda9        20161172   5524660  13612372  29% /home
/dev/hda2           62217      6843     52161  12% /rescue
/dev/hda8         5044156     42548   4745376   1% /tmp
/dev/hda6         5044156   2613132   2174792  55% /usr
none               127708         0    127708   0% /dev/shm
//minimatou/c$   14099200   9895424   4203776  71% /mnt/mmatou
lmatou:/          1554264    215884   1258056  15% /mnt/matou
lmatou:/home      2478140   1589952    760072  68% /mnt/matou/home
lmatou:/usr       1981000   1199960    678628  64% /mnt/matou/usr
lpmatou:/          995116    484112    459596  52% /mnt/pmatou
lpmatou:/home    19222656   2787880  15458228  16% /mnt/pmatou/home
lpmatou:/usr      2478140   2038764    311260  87% /mnt/pmatou/usr
deuter:/          4806936     97684   4465064   3% /mnt/deuter
deuter:/home      4806904    280100   4282620   7% /mnt/deuter/home
deuter:/files    44133352  27652876  14238608  67% /mnt/deuter/files
```

Si vous vous contentez de spécifier / dans votre liste d'inclusions, Bacula ne sauvegardera que le système de fichiers **/dev/hda5**. Pour sauvegarder tous vos systèmes de fichiers sans inclure les systèmes de fichiers montés Samba ou NFS et en excluant /tmp, /proc, .journal, et .autofsck, que vous ne voulez ni sauvegarder ni restaurer, vous pouvez utiliser ce qui suit :

```
FileSet {
  Name = Include_example
  Include {
    Options {
       wild = /proc
       wild = /tmp
       wild = \.journal
       wild = \.autofsck
       exclude = yes
```

```
    }
    File = /
    File = /boot
    File = /home
    File = /rescue
    File = /usr
  }
}
```

/tmp étant sur son propre système de fichiers et n'étant pas explicitement nommé dans la liste d'inclusion, il n'est pas nécessaire de le spécifier dans la liste d'exclusion. Cependant, il peut être préférable de le faire malgré tout par souci de clarté et au cas où il ne serait plus sur sa propre partition après un remplacement de disques.

Ayez conscience qu'il peut être **très** dangereux de permettre à Bacula de traverser ou changer de système de fichiers au gré des ppoints de montage. Par exemple, avec ce qui suit :

```
FileSet {
  Name = "Bad example"
  Include {
    Options { onefs=no }
    File = /mnt/matou
  }
}
```

vous sauvegardez une partition NFS montée (**/mnt/matou**), et puisque **onefs** est désactivée, Bacula traverse les systèmes de fichiers. Si jamais **/mnt/matou** contient lui même un point de montage où le système de fichiers de la machine sauvegardée est monté, ce qui est souvent le cas, vous vous retrouvez pris dans un boucle récursive, et la sauvegarde ne se terminera jamais.

Le FileSet suivant sauvegarde une partition raw :

```
FileSet {
  Name = "RawPartition"
  Include {
    Options { sparse=yes }
    File = /dev/hda2
  }
}
```

Lorsque vous sauvegardez et restaurez une partition raw, vous devriez vous assurer qu'aucun autre processus, y compris le système, n'écrit sur cette partition. En guise de précaution, nous recommandons ardemment de ne

sauvegarder en mode raw que des partitions non montées, ou montées en lecture seule. Ceci peut être fait si nécessaire avec la directive **RunBeforeJob**.

Considérations sur les FileSets Windows

Si vous saisissez des noms de fichiers Windows, les chemins des répertoires devraient être précédés de double-points (comme dans "c:"). Cependant, les séparateurs de champs doivent être spécifiés selon la convention Unix (c'est à dire, la barre oblique avant : "/"). Si vous souhaitez inclure une apostrophe dans un nom de fichier, précédez-la d'une barre oblique arrière (\\). Par exemple, vous pourriez utiliser ce qui suit pour sauvegarder le répertoire "My Documents" d'une machine Windows :

```
FileSet {
  Name = "Windows Set"
  Include {
    Options {
       wild = *.obj
       wild = *.exe
       exclude = yes
     }
     File = "c:/My Documents"
  }
}
```

Pour que les listes d'exclusion fonctionnent correctement sous Windows, vous devez observer les règles suivante :

- les noms de fichiers sont sensibles à la casse, aussi vous devez utilise la casse correcte ;
- vous ne devez pas spécifier de barre oblique finale pour exclure un répertoire ;
- si vos noms de fichiers comportent des espaces, vous devez les encadrer de quillemets. Les barres obliques arrières (\\) ne fonctionnent pas ;
- si vous utilisez l'ancienne syntaxe des listes d'exclusion (mentionnée ci-dessous), vous ne devriez pas spécifier la lettre référenant le disque dans une liste d'exclusion. La nouvelle syntaxe décrite ci-dessus devrait fonctionner correctement en incluant la lettre du disque.

Merci à Thiago Lima pour nous avoir résumé les points ci-dessus. Si vous rencontrez des difficultés pour faire fonctionner vos listes d'inclusion ou d'exclusion, songez à utiliser la commande **estimate job=xxx listing** documentée dans le chapitre Console chapter de ce manuel.

Sur les systèmes Win32, si vous déplacez un répertoire ou si vous renommez un fichier de la liste à sauvegarder, et si une Full a déjà eu lieu, Bacula ne saura reconnaître qu'il existe de nouveaux fichiers à sauvegarder lors d'une incrémentale ou d'une différentielle (faites-en le reproche à Microsoft, pas à moi !). Pour pallier à ce problème, veuillez copier tout répertoire ou fichier de la zone sauvegardée. Si vous ne disposez pas de suffisamment d'espace disque, déplacez-les, mais lancez alors une sauvegarde Full.

Exclusion de fichiers et répertoires

Vous pouvez aussi inclure des noms de fichiers ou chemins absolus, en plus de l'utilisation de caractères jokers et de la directive **Exclude=yes** dans les ressources Options comme exposé ci-dessus, en ajoutant simplement les fichiers à exclure dans une ressource Exclude du FileSet. Par exemple :

```
FileSet {
  Name = Exclusion_example
  Include {
    Options {
      Signature = SHA1
    }
    File = /
    File = /boot
    File = /home
    File = /rescue
    File = /usr
  }
  Exclude {
    File = /proc
    File = /tmp
    File = .journal
    File = .autofsck
  }
}
```

Un exemple de FileSet Windows

Cet exemple est une contribution de Phil Stracchino :

```
This is my Windows 2000 fileset:
FileSet {
  Name = "Windows 2000 Full Set"
  Include {
    Options {
       signature=MD5
    }
```

```
    File = c:/
  }
  Exclude {
# Most of these files are excluded not because we don't want
#  them, but because Win2K won't allow them to be backed up
#  except via proprietary Win32 API calls.
    File = "/Documents and Settings/*/Application Data/*/Profiles/
           */*/Cache/*"
    File = "/Documents and Settings/*/Local Settings/Application Data/
           Microsoft/Windows/[Uu][Ss][Rr][Cc][Ll][Aa][Ss][Ss].*"
    File = "/Documents and Settings/*/[Nn][Tt][Uu][Ss][Ee][Rr].*"
    File = "/Documents and Settings/*/Cookies/*"
    File = "/Documents and Settings/*/Local Settings/History/*"
    File = "/Documents and Settings/*/Local Settings/
           Temporary Internet Files/*"
    File = "/Documents and Settings/*/Local Settings/Temp/*"
    File = "/WINNT/CSC"
    File = "/WINNT/security/logs/scepol.log"
    File = "/WINNT/system32/config/*"
    File = "/WINNT/msdownld.tmp/*"
    File = "/WINNT/Internet Logs/*"
    File = "/WINNT/$Nt*Uninstall*"
    File = "/WINNT/Temp/*"
    File = "/temp/*"
    File = "/tmp/*"
    File = "/pagefile.sys"
  }
}
```

Remarque : les trois lignes coupées de cette liste d'exclusion ne l'ont été que pour des motifs de mise en page, elles doivent en réalité être écrites sur une seule ligne.

L'ancienne ressource FileSet

L'ancienne Ressource FileSet des versions antérieures à 1.34.3 est obsolète, mais fonctionne encore. Nous vous encourageons à utiliser la nouvelle forme, car le code correspondant sera supprimé à partir de la version 1.37.

Tester vos FileSets

Si vous voulez vous faire une idée précise de ce qui sera effectivement sauvegardé par un FileSet, ou si vous voulez vous assurer de l'efficacité d'une liste d'exclusion, vous pouvez utiliser la commande **estimate** du programme Console. Voyez estimate command dans le chapitre Console de ce manuel.

Considerations sur le nommage Windows NTFS

Les noms de fichiers NTFS contenant des caractères Unicode (i.e. > 0xFF) ne peuvent, pour le moment, être nommé eplicitement. Vous devez inclure de tels fichiers en désignant un répertoire de niveau supérieur ou une lettre de disque ne contenant pas de caractère Unicode.

La ressource Client

La ressource Client définit les attributs des clients servis (sauvegardés) par ce Director. Il faut une ressource Client par machine sauvegardée.

Client (ou FileDaemon) Début des directives Client.

Name = <name> Le nom du client qui sera utilisé dans la directive Job de la ressource ou dans une commande run de la Console. Cette directive est requise.

Address = <address> L'adresse d'un File Daemon Bacula est un nom d'hôte, un nom pleinement qualifié ou une adresse réseau au format quatre octets pointés. Cette directive est requise.

FD Port = <port-number> Où le port est le numéro de port auquel le le File Daemon peut être contacté. La valeur par défaut est 9102.

Catalog = <Catalog-resource-name> Cette directive spécifie le nom de la ressource catalog à utiliser pour ce client. Cette directive est requise.

Password = <password> Il s'agit ici du mot de passe à utiliser lors de la connection avec le File Daemon, aussi le fichier de configuration de la machine à sauvegarder doit définir le même mot de passe pour être connecté par ce Director. Cette directive est requise. Si vous disposez de **/dev/random** ou de **bc** sur votre machine, Bacula génère des mots de passe aléatoires lors du processus de configuration. Dans le cas contraire, le mot de passe est laissé blanc.

File Retention = <time-period-specification> La directive File Retention définit la période pendant laquelle Bacula conservera les enregistrements File dans le catalogue. Lors de l'expiration de cette période, si la directive **AutoPrune** est active (**yes**), Bacula élague le catalogue des enregistrements File dont l'ge est supérieur à la période de rétention. Notez que ceci n'affecte que les enregistrements du catalogue, et non vos sauvegardes archivées.

Les enregistrements File peuvent en fait être conservés pour une période inférieure à celle affectée à cette directive dans les cas où vous avez spécifié des périodes plus courtes pour les directives **Job Retention** ou **Volume Retention**. La plus courte des trois prend le

pas sur les autres. Les durées peuvent être exprimées en secondes, minutes, heures, jours, semaines, mois, trimestres ou années. Consultez le chapitre Adapter les fichiers de configuration de ce manuel pour plus de détails sur les spécifications de durées.

La valeur par défaut est de 60 jours.

Job Retention = <time-period-specification> La directive Job Retention définit la période pendant laquelle Bacula conservera les enregistrements Job dans le catalogue. Lors de l'expiration de cette période, si la directive **AutoPrune** est active (**yes**), Bacula élague le catalogue des enregistrements File dont l'ge est supérieur à la période de rétention. Notez que ceci n'affecte que les enregistrements du catalogue, et non vos sauvegardes archivées.

Si un enregistrement de Job est sélectionné pour élagage, tous les enregistrements File et JobMedia associés seront aussi élagués du catalogue, sans qu'il ne soit tenu compte de la période File Retention définie. Par conséquent, vous utiliserez, en principe, une période File Retention inférieure à la période Job retention. La période Job retention peut en fait s'avérer inférieure à la valeur que vous avez spécifiée si vous avez affecté une valeur inférieure à la directive **Volume Retention** dans la ressource Pool. Les périodes Job retention et Volume retention sont appliquées indépendamment, la plus petite prend le pas sur l'autre.

Les durées peuvent être exprimées en secondes, minutes, heures, jours, semaines, mois, trimestres ou années. Consultez le chapitre Adapter les fichiers de configuration de ce manuel pour plus de détails sur les spécifications de durées.

La valeur par défaut est 180 jours.

AutoPrune = <yes—no> Si AutoPrune est réglé à **yes** (valeur par défaut), Bacula (dans les versions au delà de 1.20) applique automatiquement les périodes File retention et Job retention pour le client à la fin du job. Si vous spécifiez **AutoPrune = no**, l'élagage ne se fera pas et votre catalogue grossira à chaque job. L'élagage n'affecte que les données du catalogue, et non les données stockées sur les volumes.

Maximum Concurrent Jobs = <number> <number> désigne le nombre maximum de jobs qui peuvent être lancés simultanément sur le client concerné. Notez que cette directive ne limite que les jobs pour les clients qui ont le même nom que la ressource dans laquelle elle apparaît. Toutes les autres restrictions du nombre maximum de jobs simultanés telles que celles spécifiées dans les ressources Director, Job, ou Storage s'appliquent aussi en plus de toute limite fixée ici. La valeur par défaut est 1, mais vous pouvez spécifier une valeur plus grande. Nous recommandons fortement de lire les MISES EN GARDE de la section Maximum Concurrent Jobs du chapitre Configurer le

Director.

Priority = <number> <number> spécifie la priorité de ce client par rapport aux autres clients que le Director sauvegarde simultanément. La priorité admet une valeur entre 1 et 1000. Les clients sont ordonnés de sorte que ceux dont la valeur de priorité sont les plus petites sont traités les premiers (Ceci n'est pas encore implémenté).

Voici un exemple d'une définition de ressource Client valide :

```
Client {
  Name = Minimatou
  Address = minimatou
  Catalog = MySQL
  Password = very_good
}
```

La ressource Storage

La ressource Storage définit les Storage Daemons disponibles pour le Director.

Storage Début des ressources Storage. Il faut en spécifier au moins une.

Name = <name> Le nom de la ressource Storage. Ce nom apparaît au niveau de la directive Storage spécifiée dans la ressource Job et est requise.

Address = <address> Où l'adresse est un nom d'hôte, une **adresse pleinement qualifiée**, ou une **adresse IP**. Notez bien que l'adresse spécifiée ici sera transmise au File Daemon qui l'utilisera alors pour contacter le Storage Daemon. Aussi, ce n'est **pas** une bonne idée d'utiliser **localhost** en tant que nom, il vaut mieux utiliser un nom de machine pleinement qualifié, ou une adresse IP. Cette directive est requise.

SD Port = <port> Où "port" est le port à utiliser pour contacter le Storage Daemon pour les informations et pour exécuter les jobs. Ce même numéro de port doit apparaître dans la ressource Storage du fichier de configuration du Storage Daemon. Le port par défaut est 9103.

Password = <password> Il s'agit du mot de passe à utiliser lors de l'établissement de la connection avec les services Storage. Ce même mot de passe doit aussi apparaître dans la ressource Director du fichier de configuration du Storage Daemon. Cette directive est requise. Si vous avez soit **/dev/random**, soit **bc** sur votre machine, Bacula génèrera un mot de passe aléatoire lors du processus de configuration. Dans le cas contraire, ce champ sera laissé blanc.

Device = <device-name> Cette directive spécifie le nom (pour le Storage Daemon) du périphérique à utiliser pour le stockage. Ce nom n'est pas le nom de périphérique physique, mais le nom de périphérique logique qui a été défini par la directive **Name** de la ressource **Device** du fichier de configuration du Storage Daemon. Si le périphérique est une librairie, vous devez utiliser le nom défini au niveau de la directive **Name** défini dans la définition de la ressource **Autochanger** du Storage Daemon. Vous pouvez utiliser le nom de votre choix (y compris le nom de périphérique physique) à concurrence de 127 caractères. Le nom de périphérique physique associé à ce périphérique est spécifié dans le fichier de configuration du Storage Daemon (en tant qu'**Archive Device**). Prenez garde à ne pas définir deux directives Storage Resource dans le Director qui pointent vers le même périphérique du Storage Daemon. Il pourrait en résulter un blocage du Storage Daemon si celui-ci tente d'utiliser le même périphérique deux fois. Cette directive est requise.

Media Type = <MediaType> Cette directive spécifie le type de média à utiliser pour stocker les données. Il s'agit d'une chaîne de caractères arbitraire n'excédant pas 127 caractères. Ce peut être ce que vous voulez, cependant, il est préférable de la choisir de manière à décrire le médium de stockage utilisé (par exemple : Fichier, DAT, "HP DLT8000", 8mm,...). D'autre part, il est esentiel d'avoir une spécification **Media Type** unique pour chaque type de média de stockage. Si vous avez deux lecteurs DDS-4 avec des formats incompatibles et une librairie DDS-4, vous devriez certainement spécifier des **Media Types** distincts. Lors d'une restauration, supposons qu'un Media Type **DDS-4** est associé avec le le job, Bacula peut décider d'utiliser tout Storage Daemon qui supporte le Media Type **DDS-4** sur tout lecteur qui le supporte.

Actuellement, Bacula n'autorise qu'un seul type de media. Par conséquent, si vous disposez d'un lecteur qui en supporte plusieurs, vous pouvez utiliser une chaîne unique pour désigner les volumes de l'un ou l'autre type, par exemple Media Type = DDS-3-4 pour les types DDS-3 et DDS-4, mais ces volumes ne seront montés que sur les lecteurs spécifiés comme acceptant les deux types : DDS-3-4

Si vous voulez contraindre Bacula à utiliser un seul Storage Daemon ou un seul lecteur, vous devez spécifier un Media Type unique pour ce lecteur. C'est un point important qui devrait être bien compris. Notez que ceci s'applique également aux volumes disque. Si vous définissez plus d'une ressource Device disque dans votre fichier de configuration du Storage Daemon, les volumes sur ces deux devices seront en fait incompatibles car l'un ne pourra être monté sur l'autre puisqu'ils se trouvent dans des répertoires différents. C'est pourquoi vous devriez

probablement plutôt utiliser deux Media Types distincts pour vos deux devices disque (même si vous pensez à eux comme ayant l'un et l'autre le type File).

Vous trouverez pus de détails à ce sujet dans le chapitre Gestion des volumes : fondements (NDT:basic volumes management) de ce manuel.

Le **MediaType** spécifié ici **doit** correspondre au **MediaType** spécifié dans la ressource **Device** du fichier de configuration du Storage Daemon. Cette directive est requise, et est utilisée par le Director et le Storage Daemon pour s'assurer qu'un volume sélectionné automatiquement dans un Pool correspond à un périphérique physique. Si un Storage Daemon gère plusieurs périphériques (par exemple, s'il écrit sur plusieurs volumes de type File sur différentes partitions), cette directive vous permet de préciser exactement quel périphérique utiliser.

Comme mentionné ci-dessus, la valeur spécifiée dans la ressource Storage du Director doit s'accorder avec celle spécifiée dans la ressource Device du fichier de configuration du Storage Daemon. Ceci représente aussi un contrôle supplémentaire pour assurer que vous n'essayez pas d'écrire les données destinées à un lecteur DLT sur un lecteur 8mm.

Autochanger = <yes—no> Si vous spécifiez **yes** pour cette commande (la valeur par défaut est **no**), alors Bacula requérira un numéro de Slot lorsque vous utiliserez les commandes **label** et **add** pour créer un nouveau volume. Ceci simplifie la création des enregistrements de Volumes dans le catalogue si vous disposez d'une librairie. Si vous omettez de spécifier le Slot, la robotique ne sera pas utilisée. Cependant, vous pouvez modifier le Slot associé à un volume à tout moment grce à la commande **update volume** du programme Console. Lorqu'**autochanger** est activée, l'algorithme utilisé par Bacula pour rechercher des volumes utilisables est modifié de faon à prendre en compte tout volume supposé contenu dans la librairie. Si aucun volume **in changer** n'est trouvé, Bacula tente de recycler, élaguer..., et s'il ne trouve toujours aucun volume, Bacula recherche un volume présent ou non dans la librairie. Privilégier les volumes présents dans la librairie permet de minimiser les interventions d'un opérateur.

Pour que la robotique soit utilisée, vous devez aussi spécifier **Autochanger = yes** dans la ressource La Ressource Device du fichier de configuration du Storage Daemon, ainsi que d'autres paramètres importants du Storage Daemon. Vous trouverez plus d'information à ce sujet dans le chapitre Utiliser une librairie de ce manuel.

Maximum Concurrent Jobs = <number> Où <number> est le nombre maximum de jobs utilisant la ressource Storage courante qui peuvent être exécutés simultanément. Notez que cette directive ne limite que les jobs qui utilisent ce Storage Daemon. Toute autre

limitation du nombre maximum de jobs simultanés au niveau des ressources Director, Job ou Client est aussi appliquée en plus de celle fixée ici. La valeur par défaut est 1, mais vous pouvez utiliser une valeur plus importante. Nous vous recommandons fortement de lire les MISES EN GARDE de la section Maximum Concurrent Jobs du chapitre Configurer le Director.

Alors qu'il est possible de définir des nombres maximum de jobs simultanés supérieurs à 1 dans les ressource Director, Job et Client, vous devriez porter une attention particulière au paramétrage de cette directive pour le Storage Daemon. En conservant la valeur 1, vous évitez que deux jobs écrivent simultanément sur le même volume ce qui, quoique supporté, n'est pas recommandé actuellement.

Voici un exemple de ressource Storage valide :

```
# Definition of tape storage device
Storage {
  Name = DLTDrive
  Address = lpmatou
  Password = storage_password # password for Storage daemon
  Device = "HP DLT 80"    # same as Device in Storage daemon
  Media Type = DLT8000    # same as MediaType in Storage daemon
}
```

La ressource Pool

La ressource Pool définit l'ensemble des volumes de stockage (cartouches ou fichiers) à la disposition de Bacula pour écrire les données. En configurant différents Pools, vous pouvez déterminer quel ensemble de volumes (ou média) reoit les données sauvegardées. Ceci permet, par exemple, de stocker toutes les sauvegardes Full sur un ensemble de volumes, et les sauvegardes différentielles et incrémentales sur un autre. De même, vous pouvez assigner un ensemble de volumes à chaque machine sauvegardée. Tout ceci peut être réalisé aisément en définissant plusieurs pools.

Un autre aspect important d'un pool est qu'il contient des attributs par défaut (Nombre maximum de jobs, période de rétention, drapeau de recyclage,...) qui sont conférés à tout volume lui appartenant lors de sa création. Ceci vous évite d'avoir à répondre à un grand nombre de questions lorsque vous étiquettez (label) un nouveau volume. Chacun de ces attributs peut ensuite être modifié sur chaque volume individuellement avec la commande **update** du programme Console. Notez que vous devez préciser explicitement quel pool est à utiliser avec chaque job. Bacula ne recherche pas automatiquement le pool correct.

Dans la plupart des installations de Bacula, toutes les sauvegardes de toutes les machines vont vers un unique jeu de volumes. Dans ce cas, vous n'utiliserez probablement que le pool par défaut **Default**. Si votre stratégie de sauvegarde vous impose à monter chaque jour une cartouche différente, vous voudrez probablement définir des pools distincts pour chaque jour. Pour plus d'informations à ce sujet, consultez le chapitre Stratégies de sauvegarde de ce manuel.

Pour utiliser un pool, vous devez suivre trois étapes :

D'abord, le pool doit être défini dans le fichier de configuration du Director. Ensuite le pool doit être enregistré dans le catalogue. Ceci est fait automatiquement par le Director à chaque fois qu'il démarre, ou peut être réalisé manuellement à l'aide de la commande **create** du programme Console. Enfin, si vous modifiez la définition du pool dans le fichier de configuration du Director et redémarrez Bacula, le pool sera mis à jour automatiquement, ce qui peut aussi être réalisé manuellement avec la commande **update pool** du programme Console pour rafraichir l'image du catalogue. C'est cette image du catalogue plutôt que l'image de la ressource du Director qui est utilisée pour les attributs de volume par défaut. Notez que pour que le pool soit automatiquement créé ou mis à jour, il doit être référencé explicitement par une ressource Job.

Ensuite le médium physique doit être étiquetté. L'étiquettage peut être réalisé soit par la commande **label** du programme Console, soit en utilisant le programme **btape**. La méthode à privilégier est la première.

Finallement, vous devez ajouter des noms de volumes (et leurs attributs) au pool. Pour que les volumes soient utilisés par Bacula, ils doivent être du même **Media Type** que l'Archive Device spécifiée pour le job. (Autrement dit, si vous vous apprétez à sauvegarder vers un lecteur DLT, le pool doit contenir des volumes DLT, puisque des volumes 8mm ne peuvent être montés sur un lecteur DLT). Le **Media Type** revêt une importance particulière si vous sauvegardez vers des fichiers. Lorsque vous exécutez un job, vous devez explicitement préciser le pool. Bacula sélectionne dès lors automatiquement le prochain volume du pool à utiliser, en s'assurant que le **Media Type** de tout volume sélectionné est bien celui requis par la ressource Storage spécifiée pour le job.

Si vous utilisez la commande **label** du programme Console pour étiquetter les volumes, il sont automatiquement ajoutés au pool, aussi cette dernière étape n'est généralement pas requise.

Il est aussi possible d'ajouter des volumes au catalogue sans avoir explicitement étiquetté les volumes physiques. Ceci s'effectue avec la commande

add du programme Console.

Comme mentionné plus haut, à chaque démarrage, Bacula examine tous les pools associés à chaque catalogue, et si un enregistrement n'existe pas encore, il est créé à partir de la définition du pool dans la ressource. Bacula devrait probablement effectuer un **update pool** si vous modifiez la définition du pool mais, actuellement, vous devez le faire manuellement avec la commande **update pool** du programme Console.

La ressource Pool définie dans le fichier de configuration du Director peut contenir les directives suivantes :

Pool Début de la ressource Pool. Il faut définir au moins une ressource Pool.

Name = <name> Le nom du pool. Pour la plupart des applications, vous utiliserez le pool par défaut nommé **Default**. Cette directive est requise.

Number of Volumes = <number> Cette directive spécifie le nombre de volumes (cartouches ou fichiers) contenus dans le pool. En principe, il est défini et mis à jour automatiquement par la routine de maintenance du catalogue de Bacula.

Maximum Volumes = <number> Cette directive spécifie le nombre maximum de volumes contenus dans le pool. Cette directive est optionnelle. Si elle est omise ou réglée à 0, tout nombre de volumes est permis. En général, cette directive est utile pour les librairies, où il y a un nombre fixé de volumes, ou pour le stockage sur fichier si vous voulez vous assurer que les sauvegardes sur disque ne deviennent pas trop nombreuses ou ne consomment pas trop d'espace.

Pool Type = <type> Cette directive définit le type du pool, qui correspond au type du job exécuté. Cette directive est requise et peut prendre l'une des valeurs suivantes :

Backup
*Archive
*Cloned
*Migration
*Copy
*Save

Use Volume Once = <yes—no> Cette directive, si elle est active (valeur **yes**) stipule que chaque volume ne doit être utilisé qu'une seule fois. C'est particulièrement utile si le volume est un fichier et si vous voulez un nouveau fichier pour chaque nouvelle sauvegarde. La valeur

par défaut est **no** (autrement dit, les volumes peuvent être utilisés plusieurs fois). Cette directive sera très certainement bientôt obsolète, aussi nous vous recommandons d'utiliser **Maximum Volume Jobs = 1** à la place.

La valeur définie par cette directive dans le fichier de configuration du Director est la valeur par défaut utilisée lorsqu'un nouveau volume est créé. Modifier la valeur dans le fichier de configuration ne changera pas ce qui est stocké sur le volume. Pour changer cette valeur pour un volume existant, vous devez utiliser la commande **update** du programme Console.

Maximum Volume Jobs = <positive-integer> Cette directive spécifie le nombre maximum de jobs qui peuvent être écrits sur le volume. La valeur par défaut est zéro, ce qui signifie que le nombre de jobs sur un volume n'est pas limité. Sinon, lorsque le nombre de jobs sauvegardés sur le volume atteint la valeur **positive-integer** spécifiée, le volume est marqué **Used**. Dès lors, il ne peut plus être utilisé pour y ajouter des jobs, tout comme dans le cas d'un volume avec le statut **Full**, mais il peut être recyclé si le recyclage est activé. En spécifiant **MaximumVolumeJobs = 1** vous obtenez le même effet que celui que vous obtiendriez avec **UseVolumeOnce = yes**.

Notez que la valeur définie par cette directive dans le fichier de configuration du Director est la valeur par défaut utilisée lors de la création d'un volume. Une fois le volumes créé, modifier la valeur dans le fichier de configuration ne changera pas ce qui est stocké sur le volume. Pour changer cette valeur pour un volume existant, vous devez utiliser la commande **update** du programme Console.

Maximum Volume Files = <positive-integer> Cette directive spécifie le nombre maximum de fichiers qui peuvent être écrits sur le volume. La valeur par défaut est zéro, ce qui signifie que le nombre de fichiers sur un volume n'est pas limité. Sinon, lorsque le nombre de fichiers sauvegardés sur le volume atteint la valeur **positive-integer** spécifiée, le volume est marqué **Used**. Dès lors, il ne peut plus être utilisé pour y ajouter des jobs, tout comme dans le cas d'un volume avec le statut **Full**, mais il peut être recyclé si le recyclage est activé. Ce n'est qu'à la fin d'un job que Cette valeur est controlée, et que le statut du volume est éventuellement changé en **Used**.

La valeur définie par cette directive dans le fichier de configuration du Director est la valeur par défaut utilisée lors de la création d'un volume. Une fois le volumes créé, modifier la valeur dans le fichier de configuration ne changera pas ce qui est stocké sur le volume. Pour changer cette valeur pour un volume existant, vous devez utiliser la commande **update** du programme Console.

Maximum Volume Bytes = <size> Cette directive spécifie le nombre

maximum d'octets qui peuvent être écrits sur le volume. La valeur par défaut est zéro, ce qui signifie qu'il n'y a d'autre limite que la capacité physique du volume. Sinon, lorsque le nombre d'octets sauvegardés sur le volume atteint la valeur **size** spécifiée, le volume est marqué **Used**. Dès lors, il ne peut plus être utilisé pour y ajouter des jobs, tout comme dans le cas d'un volume avec le statut **Full**, mais il peut être recyclé si le recyclage est activé. Ce n'est qu'à la fin d'un job que Cette valeur est controlée, et que le statut du volume est éventuellement changé en **Used**.

La valeur définie par cette directive dans le fichier de configuration du Director est la valeur par défaut utilisée lors de la création d'un volume. Une fois le volumes créé, modifier la valeur dans le fichier de configuration ne changera pas ce qui est stocké sur le volume. Pour changer cette valeur pour un volume existant, vous devez utiliser la commande **update** du programme Console.

Volume Use Duration = <time-period-specification> Cette directive définit la période durant laquelle le volume peut être utilisé en écriture, à compter de la première opération d'écriture de données. La valeur par défaut est zéro, ce qui signifie que le volume peut être écrit indéfiniment . Sinon, lorsque la durée depuis la première écriture excède la période **time-period-specification** spécifiée, le volume est marqué **Used**. Dès lors, il ne peut plus être utilisé pour y ajouter des jobs, tout comme dans le cas d'un volume avec le statut **Full**, mais il peut être recyclé si le recyclage est activé. L'usage de la commande **status dir** applique des algorithmes similaires à l'éxecution de jobs, aussi lors d'une telle commande, le statut du volume peut être modifié. Lorsque le volume est recyclé, il peut à nouveau être utilisé.

Vous pourriez utiliser cette directive, par exemple, si vous avez un volume dédié aux sauvegardes incrémentales, et un volume dédié aux Fulls hebdomadaires. Une fois que la Full est passée, vous pouvez préférer utiliser un autre volume pour les incrémentales. Ceci peut être accompli en règlant la période **Volume Use Duration** à six jours pour les volumes des incrémentales. Autrement dit, celui-ci sera utilisé durant les six jours qui suivent une full, puis un autre volume d'incrémentales sera utilisé. Veillez à utiliser des périodes relativement courtes telles que 23 heures, ou vous pourriez placer Bacula en situation de devoir attendre tout un week-end le montage d'une cartouche par l'opérateur pour pouvoir terminer une sauvegarde.

Ce n'est qu'à la fin d'un job que Cette valeur est controlée, et que le statut du volume est éventuellement changé en **Used**, ce qui signifie que bien que la période **Volume Use Duration** puisse avoir expiré, l'entrée correspondante du catalogue ne sera pas mise à jour jusqu'à ce que le prochain job utilisant ce volume soit exécuté.

La valeur définie par cette directive dans le fichier de configuration du Director est la valeur par défaut utilisée lors de la création d'un volume. Une fois le volumes créé, modifier la valeur dans le fichier de configuration ne changera pas ce qui est stocké sur le volume. Pour changer cette valeur pour un volume existant, vous devez utiliser la commande **update volume** du programme Console.

Catalog Files = <yes—no> Cette directive précise si les noms des fichiers sauvegardés doivent ou non être enregistrés dans le catalogue. La valeur par défaut est **yes**. L'avantage de désactiver cette option (**Catalog Files = No**) est d'avoir un catalogue significativement plus petit. L'inconvénient est de ne pas pouvoir produire de liste des fichiers sauvegardés pour chaque job à partir du catalogue (autrement dit, vous ne pourrez naviguer dans les fichiers sauvegardés). Ainsi, sans les enregistrements de fichiers, vous ne pourrez utiliser la commande **restore**, ni aucune des autres commandes du programme Console qui se réfèrent aux enregistrements de fichiers.

AutoPrune = <yes—no> Si AutoPrune est activée (**yes**), ce qui est le cas par défaut, Bacula (depuis la version 1.20) applique automatiquement la période de rétention des volumes lorsqu'un nouveau volumes est requis ou lorsqu'il n'existe aucun volume utilisable dans le pool. L'élagage des volumes consiste à supprimer du catalogue les jobs expirés (ceux qui sont plus anciens que la période de rétention), et rend possible le recyclage de ces volumes

Volume Retention = <time-period-specification> La directive Volume Retention définit la période durant laquelle **Bacula** conserve les enregistrements Job associés au volume dans le catalogue. Lors de l'expiration de cette période, si **AutoPrune** est activée, Bacula peut supprimer les enregistrements Job plus anciens que la période **time-period-specification** spécifiée s'il est nécessaire de libérer un volume. Tous les enregistrements de fichiers associés à des jobs supprimés sont aussi supprimés. Les durées peuvent être exprimées en secondes, minutes, heures, jours, semaines, mois, trimestres ou années. La période **Volume Retention** s'applique indépendamment des périodes **Job Retention** et **File Retention** définies dans la ressource Client. Ceci signifie que la période la plus courte est celle qui s'applique.Notez bien que lorsque la période **Volume Retention** a été atteinte les enregistrements de Job et ceux de fichiers sont supprimmés les uns et les autres. Cet élagage peut aussi se produire à l'exécution de la commande **status dir** car elle applique des algorithmes similaires à ceux utilisés pour déterminer le prochain volume disponible.

Il est important de savoir que lorsque la période Volume Retention expire, Bacula ne recycle pas systématiquement le volume concerné. Il tente de conserver les données intactes aussi longtemps que possible

avant d'écrire sur ce volume.

La valeur par défaut est 365 jours. Notez que cette directive règle la valeur par défaut pour chaque enregistrement de volume du catalogue lorsque le volume est créé. Cette valeur du catalogue peut ensuite être modifiée avec la commande **update** du programme Console.

En définissant plusieurs pools avec différentes périodes de rétention (volume retention), vous pouvez efficacement gérer vos cartouches avec, par exemple un pool de cartouches recyclé chaque semaine, un autre recyclé chaque mois, et ainsi de suite. Cependant, il faut bien garder à l'esprit que si votre période de rétention **Volume Retention** est trop courte, il peut arriver que votre dernière sauvegarde Full valide soit supprimée, de sorte que vous n'ayez plus une sauvegarde complète de votre système, et que votre prochaine incrémentale soit élevée en une Full. Par conséquent, la valeur minimum de la période **Volume Retention** devrait être au moins le double de l'intervalle séparant vos Fulls. Autrement dit, pour des Fulls mensuelles, la période **Volume Retention** devrait être supérieure ou égale à deux mois.

Notez que la valeur définie par cette directive dans le fichier de configuration du Director est la valeur par défaut utilisée lors de la création d'un volume. Une fois le volumes créé, modifier la valeur dans le fichier de configuration ne changera pas ce qui est stocké sur le volume. Pour changer cette valeur pour un volume existant, vous devez utiliser la commande **update** du programme Console.

Recycle = <yes—no> Cette directive spécifie la valeur par défaut pour le recyclage des volumes purgés. Si elle est activée (**yes**) et si Bacula a besoin d'un volume mais n'en trouve aucun utilisable, il recherche alors les volumes purgés (c'est à dire ceux dont tous les jobs et fichiers ont expiré et ont, de fait, été supprimés du catalogue). Si un volume est recyclé, toutes les données précedemment écrites sur ce volumes seront écrasées. Si le recyclage est désactivé (**no**), le volume ne sera pas recyclé, et ainsi les données resteront valides. Si vous souhaitez réutiliser un volume dont le drapeau de recyclage est no (0 dans le catalogue), vous devez manuellement le changer en yes (commande update).

Notez que la valeur définie par cette directive dans le fichier de configuration du Director est la valeur par défaut utilisée lors de la création d'un volume. Une fois le volumes créé, modifier la valeur dans le fichier de configuration ne changera pas ce qui est stocké sur le volume. Pour changer cette valeur pour un volume existant, vous devez utiliser la commande **update** du programme Console.

Recycle Oldest Volume = <yes—no> Cette directive indique au Director de rechercher le volume le plus ancien du pool lorsq'un nouveau est requis par le Storage Daemon et qu'aucun autre n'est disponible.

Le catalog est alors **élagué** dans le respect des périodes **Job**, **File** et **Volume Retention**, c'est pourquoi cette directive est, de **très** loin, préférable à directive Purge Oldest Volume.

Cette directive peut être utile si vous avez un nombre fixe de volumes dans un pool et que vous voulez cycler sur ces volumes après avoir spécifié les périodes de rétention qui conviennent.

Recycle Current Volume = <yes—no> Si Bacula a besoin d'un nouveau volume, cette directive lui indique d'élaguer le volume dans le respect des périodes **Job**, **File** et **Volume Retention**. Si tous les jobs sont élagués, (c'est à dire si le volume est purgé), alors le volume est recyclé et sera utilisé pour la prochaine opération d'écriture. Cette directive respecte toutes les périodes **Job**, **File** et **Volume Retention**, c'est pourquoi elle est , de **très** loin, préférable à la directive Purge Oldest Volume.

Cette directive peut être utile si vous avez un nombre fixe de volumes dans un pool et que vous voulez cycler sur ces volumes après avoir spécifié les périodes de rétention qui élaguent les volumes avant que vous n'ayez terminé le cycle sur les volumes.

Purge Oldest Volume = <yes—no> Cette directive indique au Director de rechercher le volume utilisé le plus ancien dans le pool lorsqu'un nouveau volume est requis par le Storage Daemon et qu'aucun n'est disponible. Le catalogue est alors purgé sans égards pour les périodes de rétention des fichiers et jobs écrits sur ce volume. Le volume est alors recyclé pour être utilisé pour la prochaine opération d'écriture. Cette directive outrepasse toute période de rétention (Job, File, ou Volume) que vous avez pu spécifier par ailleurs.

Cette directive peut être utile si vous avez un nombre fixe de volumes dans un pool et que vous voulez cycler sur ces volumes lorsque tous les volumes sont pleins, sans avoir à vous soucier de paramétrer les périodes de rétentions qui conviendraient. Cependant, en l'utilisant, vous courrez le risque de perdre toutes vos données.

Soyez conscient que Purge Oldest Volume ne fait aucun cas d'aucune période de rétention. Si vous activez cette directive alors que vous ne possédez qu'un seul volume, ce volume sera systématiquement écrasé tout de suite après avoir été rempli ! Aussi, assurez vous au moins d'avoir un nombre décent de volumes dans votre pool avant d'exécuter un job. Si vous voulez que les périodes de rétention soient prises en compte, n'utilisez pas cette directive. Pour spécifier une période de rétention, utilisez la directive **Volume Retention** (voir ci dessus).

Je recommande fortement de ne pas utiliser cette directive, car il est certain que tôt ou tard, Bacula recyclera un volume contenant des données valides et récentes. La valeur par défaut est **no**

Cleaning Prefix = <string> Cette directive définit un préfixe qui, s'il correspond au début du nom d'un volume lors de son étiquettage, indique à Bacula que le volume en question est une cartouche de nettoyage, qui aura le statut **VolStatus = Cleaning**. Ainsi Bacula ne tentera jamais d'utiliser cette cartouche. Cette option est particulièrement utile avec les librairies équipées de lecteurs de codes barres où, conventionnellement, les codes barres commenant par **CLN** sont traités en tant que cartouches de nettoyage.

Label Format = <format> Cette directive précise le format des étiquettes des volumes de ce pool. La directive **Format** est utilisée comme un patron pour créer de nouveaux noms de volumes lors de l'étiquettage automatique.

Le **format** devrait être spécifié entre guillemets, et consiste en une chaîne de lettres, chiffres et caractères spéciaux tiret (**-**), souligné (**_**), double point (**:**) et point (**.**), qui sont considérés valides pour un nom de volume.

De plus, le **format** peut comporter des caractères variables qui seront substituées par un algorithme complexe, ce qui permet de créer des noms de volumes avec plusieurs formats différents. En tous les cas, le processus d'expansion des variables doit aboutir au jeu de caractères définis comme légaux dans le dernier paragraphe. Généralement, ces caractères variables commencent par un signe dollar (**$**) ou un crochet droit ([). Si vous spécifiez des caractères variables vous devriez toujours les encadrer de guillemets. Pour plus de détails sur ce sujet, veuillez consulter le chapitre Variable Expansion de ce manuel.

Si aucun caractère variable n'est découvert dans la chaîne, le nom de volume sera constitué de la chaîne **format** suffixée du nombre de volumes dans le pool plus un, au format 4 chiffres et avec des zéros en tête. Par exemple, avec **Label Format = "File-"**, les volumes seront nommés **File-0001**, **File-0002**, ...

Exception faite des variables spécifiques aux jobs, vous pouvez tester votre **LabelFormat** en utilisant la section var command du chapitre Console de ce manuel.

Dans la plupart des cas, vous devriez encadrer la spécification de format (la partie à droite du signe égale) entre guillemets. Notez que cette directive est obsolète et qu'elle est remplacée, à partir de la version 1.37 par un script Python pour la création des noms de volumes.

Pour qu'un pool puisse être utilisé lors d'une sauvegarde, il faut qu'il lui soit associé au moins un volume. Les volumes sont créés et affectés aux pools avec les commandes **label** ou **add** du programme **Bacula Console**. Outre l'affectation du volume au pool (c'est à dire son référencement dans le catalogue), le volume physique doit recevoir une étiquette logicielle valide

pour que Bacula l'accepte. Ceci peut être réalisé automatiquement grce à la commande **label**. D'autre part, Bacula peut effectuer cette opération automatiquement si l'instruction lui en est donné, mais cette fonctionnalité n'est pas encore pleinement implémentée.

Voici un exemple d'une définition de ressource Pool valide :

```
Pool {
  Name = Default
  Pool Type = Backup
}
```

Le Scratch Pool

En général, vous pouvez nommer vos pool à votre guise, il existe cependant une importante restriction : le pool nommé **Scratch**, s'il existe, se comporte comme une réserve de volumes où Bacula pourra puiser s'il ne trouve aucun volume utilisable dans le pool normalement utilisé par le job. Le volume est alors déplacé du pool Scratch vers le pool en défaut.

La ressource Catalog

La ressource Catalog précise quel catalogue utiliser pour le job courant. Actuellement, Bacula ne peut utiliser qu'un type de serveur de bases de données défini lors de sa configuration : SQLite, MySQL, PostgreSQL. En revanche, vous pouvez utiliser autant de catalogues que vous le souhaitez. Par exemple, vous pouvez avoir un catalogue par client, ou encore un catalogue pour les sauvegardes, un autre pour les jobs de type Verify et un troisième pour les restaurations.

Catalog Début de la ressource Catalog. Il faut au moins une ressource Catalogue.

Name = <name> Le nom du Catalog. Il n'a pas besoin d'être en relation avec le nom sur le serveur de base de données. Ce nom sera repris dans la ressource Client, indiquant ainsi que toutes les données relatives à ce client sont maintenues dans ce catalogue. Cette directive est requise.*

password = <password> Cette directive spécifie le mot de passe à utiliser pour se connecter au catalogue. Cette directive est requise.

DB Name = <name> Cette directive spécifie le nom de la base de données. Si vous utilisez plusieurs catalogues, vous spécifiez lequel ici. Si vous utilisez un serveur de bases de données externe plutôt que

l'intégré, vous devez spécifier un nom connu du serveur (autrement dit, le nom que vous avez utilisé lorsque vous avez créé les tables Bacula.). Cette directive est requise.

user = <user> L'utilisateur habilité à se connecter au catalogue. Cette directive est requise.

DB Socket = <socket-name> Il s'agit du nom d'un *socket* à utiliser sur la machine locale pour se connecter au catalogue. Cette directive n'est utilisée que par MySQL, elle est ignorée par SQLite. Normalement, si ni **DB Socket**, ni **DB Address** ne sont spécifiées, MySQL utilise le socket par défaut.

DB Address = <address> Il s'agit de l'adresse du serveur de bases de données. En principe, vous utiliserez cette directive plutôt que **DB Socket** si le serveur de bases de données est une autre machine. Dans ce cas, vous spécifierez aussi le port **DB Port**. Cette directive n'est utilisée que par MySQL, et ignorée par SQLite. Elle est optionnelle.

DB Port = <port> Cette directive définit le port à utiliser en conjonction avec **DB Address** pour accéder au catalogue s'il est hébergé sur une autre machine. Elle n'est utilisée que par MySQL, et ignorée par SQLite. Elle est optionnelle.

Voici un exemple d'une définition de ressource Catalog valide :

```
Catalog
{
  Name = SQLite
  dbname = bacula;
  user = bacula;
  password = ""                         # no password = no security
}
```

en voici une deuxième pour un catalogue sur une autre machine :

```
Catalog
{
  Name = MySQL
  dbname = bacula
  user = bacula
  password = ""
  DB Address = remote.acme.com
  DB Port = 1234
}
```

*La ressource Messages

Pour les détails sur la ressource Messages, veuillez consulter le chapitre La ressource Messages de ce manuel.

La ressource Console

A partir de la version 1.33 de Bacula, l'administrateur dispose de trois sortes de consoles différentes pour interagir avec le Director. Ces trois types de consoles comportent trois niveaux de sécurité différents.

- Le premier type de console est une console **anonyme** ou **par défaut**, qui détient tous les privilèges. La ressource Console n'est pas requise pour ce type puisque le mot de passe est spécifié dans la ressource Director et par conséquent, de telles consoles n'ont pas de nom défini par une directive **Name =**. C'est le premier type de console implémenté dans les versions antérieures à 1.33, il demeure valide. Sont usage est typiquement réservé aux administrateurs.
- Le second type de console, apparu avec la version 1.33 est une console "nommée" définie dans une ressource Console simultanément dans le fichier de configuration du Director et dans le fichier de configuration de la Console, les noms et mots de passe devant être en conjonction dans ces deux fichiers. Ce type de console ne détient absolument aucun privilège, exceptés ceux explicitement spécifiés dans la ressource Console du fichier de configuration du Director. Ainsi, vous pouvez définir plusieurs consoles avec des noms et mots de passe distincts destinées à différents utilisateurs avec différents privilèges. Par défaut ces consoles ne peuvent absolument rien faire. Il vous revient de leur accorder des privilèges ou plutôt des accès aux commandes et ressources en spécifiant des listes de contrôle d'accès (ACL) dans la ressource Console du fichier de configuration du Director. Les ACLs sont spécifiées par une directive suivie d'une liste de noms d'accès. Vous trouverez des exemples ci-dessous.
- Le troisième type de console est similaire au second en ce qu'il requiert une définition de ressource Console dans le fichier de configuration du Director et dans le fichier de configuration de la Console. De plus, si le nom de la console spécifié par la directive **Name =** est le même que le nom du client, cette console est autorisée à utiliser la commande **SetIP** pour modifier la directive Address dans la ressource client du fichier de configuration du Director en l'adresse IP de la console. Ceci permet aux portables et autres machines utilisant DHCP (adresses IP dynamiques) de "notifier" le Director de leur adresse IP courante.

La ressource Console est optionnelle. Les directives suivantes sont permises dans le fichier de configuration du Director.

Name = <name> Le nom de la console. Ce nom doit être en conjonction avec celui spécifié dans le fichier de configuration de la Console (comme c'est le cas pour les définitions de clients).

Password = <password> Spécifie le mot de passe qu'une console nommée doit fournir pour être autorisée. Le même mot de passe doit apparaître dans la ressource Console du fichier de configuration de la Console. Pour plus de sécurité, le mot de passe n'est jamais réellement transmis à travers le réseau, mais plutôt un code de hachage de type *challenge response* créé à partir du mot de passe. Cette directive est requise. Si vous disposez de **/dev/random** ou **bc** sur votre machine, Bacula génèrera aléatoirement ce mot de passe lors du processus de configuration, autrement, il sera laissé blanc.

JobACL = <name-list> Cette directive est utilisée pour spécifier une liste de noms de ressources Job qui peuvent être accédés par la console. Sans cette directive, la console ne peut accéder à aucune des ressources Job définies dans le fichier de configuration du Director. Plusieurs ressources Job peuvent être spécifiées en les séparant par des virgules, et/ou en spécifiant plusieurs directives JobACL. Par exemple, la directive peut être spécifiée ainsi :

```
JobACL = kernsave, "Backup client 1", "Backup client 2"
JobACL = "RestoreFiles"
```

Avec cette spécificaton, la console peut accéder aux ressources du Director pour les quatre jobs désignés par la directive JobACL, et uniquement à eux.

ClientACL = <name-list> Cette directive est utilisée pour spécifier une liste de noms de ressources Client iaccessibles par la console.

StorageACL = <name-list> Cette directive est utilisée pour spécifier une liste de noms de ressources Storage accessibles par la console.

ScheduleACL = <name-list> Cette directive est utilisée pour spécifier une liste de noms de ressources Schedule accessibles par la console.

PoolACL = <name-list> Cette directive est utilisée pour spécifier une liste de noms de ressources Pool accesibles ar la console.

FileSetACL = <name-list> Cette directive est utilisée pour spécifier une liste de noms de ressources FileSet accessibles par la console.

CatalogACL = <name-list> Cette directive est utilisée pour spécifier une liste de noms de ressources Catalog accessibles par la console.

CommandACL = <name-list> Cette directive est utilisée pour spécifier une liste de commandes de la console qui peuvent être exécutées par la console.

En plus des différents noms de ressources du Director et de commandes, le mot-clef spécial ***all*** peut être spécifié dans chacune des directives ACL ci-

dessus. Sa présence signifie que toute ressource ou commande (pourvu qu'elle soit appropriée à la directive) est acceptée. Pour un exemple de fichier de configuration, voyez le chapitre Configuration de la console de ce manuel

La ressource Counter

La ressource Counter définit une variable-compteur qui peut être accédée par le processus d'expansion de variables utilisé pour la création de d'étiquettes (labels) de volumes avec la directive **LabelFormat**. Consultez le paragraphe sur la directive LabelFormat de ce chapitre pour plus de détails.

Counter Début de la ressource Counter. les directives Counter sont optionnelles.

Name = <name> iLe nom de la variable-compteur. Il s'agit du nom que vous utiliserez pour vous référer à cette variable et accéder à sa valeur.

Minimum = <integer> Spécifie la valeur minimale de la variable-compteur. Cette valeur devient celle par défaut. Si elle n'est pas fournie, sa valeur est zéro.

Maximum = <integer> Spécifie la valeur maximale de la variable-compteur. Si elle n'est pas fournie, ou si elles est réglée à zéro, la variable compteur peut prendre une valeur maximale de 2 147 483 648 (2^{31}). Au delà de cette valeur, le compteur est remis au minimum.

***WrapCounter = <counter-name>** Si cette valeur est spécifiée, lorsque la variable-compteur dépasse le maximum et est remise au minimum, le compteur spécifié par la directive **WrapCounter** est incrémenté. (Ceci n'est, pour le moment, pas implémenté.)

Catalog = <catalog-name> Si cette directive est spécifiée, le compteur et sa valeur sont sauvegardés dans le catalogue spécifié. Dans le cas contraire, le compteur est redéfini à chaque démarrage de Bacula.

Voici un exemple complet de fichier de configuration du Director.

Un exemple de fichier de configuration du Director pourrait être le suivant :

```
#
# Default Bacula Director Configuration file
#
#  The only thing that MUST be changed is to add one or more
#   file or directory names in the Include directive of the
#   FileSet resource.
```

```
#
#  For Bacula release 1.15 (5 March 2002) -- redhat
#
#  You might also want to change the default email address
#   from root to your address.  See the "mail" and "operator"
#   directives in the Messages resource.
#
Director {                            # define myself
  Name = rufus-dir
  QueryFile = "/home/kern/bacula/bin/query.sql"
  WorkingDirectory = "/home/kern/bacula/bin/working"
  PidDirectory = "/home/kern/bacula/bin/working"
  Password = "XkSfzu/Cf/wX4L8Zh4G4/yhCbpLcz3YVdmVoQvU3EyF/"
}
# Define the backup Job
Job {
  Name = "NightlySave"
  Type = Backup
  Level = Incremental                 # default
  Client=rufus-fd
  FileSet="Full Set"
  Schedule = "WeeklyCycle"
  Storage = DLTDrive
  Messages = Standard
  Pool = Default
}
Job {
  Name = "Restore"
  Type = Restore
  Client=rufus-fd
  FileSet="Full Set"
  Where = /tmp/bacula-restores
  Storage = DLTDrive
  Messages = Standard
  Pool = Default
}

# List of files to be backed up
FileSet {
  Name = "Full Set"
  Include {
    Options { signature=SHA1 }
#
#  Put your list of files here, one per line or include an
#    external list with:
#
#    @file-name
#
#  Note: / backs up everything
  File = /
  }
  Exclude { }
}
# When to do the backups
```

```
Schedule {
  Name = "WeeklyCycle"
  Run = Full sun at 1:05
  Run = Incremental mon-sat at 1:05
}
# Client (File Services) to backup
Client {
  Name = rufus-fd
  Address = rufus
  Catalog = MyCatalog
  Password = "MQk6lVinz4GG2hdIZk1dsKE/LxMZGo6znMHiD7t7vzF+"
  File Retention = 60d      # sixty day file retention
  Job Retention = 1y        # 1 year Job retention
  AutoPrune = yes           # Auto apply retention periods
}
# Definition of DLT tape storage device
Storage {
  Name = DLTDrive
  Address = rufus
  Password = "jMeWZvfikUHvt3kzKVVPpQ0ccmV6emPnF2cPYFdhLApQ"
  Device = "HP DLT 80"      # same as Device in Storage daemon
  Media Type = DLT8000      # same as MediaType in Storage daemon
}
# Definition for a DLT autochanger device
Storage {
  Name = Autochanger
  Address = rufus
  Password = "jMeWZvfikUHvt3kzKVVPpQ0ccmV6emPnF2cPYFdhLApQ"
  Device = "Autochanger"    # same as Device in Storage daemon
  Media Type = DLT-8000     # Different from DLTDrive
  Autochanger = yes
}
# Definition of DDS tape storage device
Storage {
  Name = SDT-10000
  Address = rufus
  Password = "jMeWZvfikUHvt3kzKVVPpQ0ccmV6emPnF2cPYFdhLApQ"
  Device = SDT-10000        # same as Device in Storage daemon
  Media Type = DDS-4        # same as MediaType in Storage daemon
}
# Definition of 8mm tape storage device
Storage {
  Name = "8mmDrive"
  Address = rufus
  Password = "jMeWZvfikUHvt3kzKVVPpQ0ccmV6emPnF2cPYFdhLApQ"
  Device = "Exabyte 8mm"
  MediaType = "8mm"
}
# Definition of file storage device
Storage {
  Name = File
  Address = rufus
  Password = "jMeWZvfikUHvt3kzKVVPpQ0ccmV6emPnF2cPYFdhLApQ"
  Device = FileStorage
```

```
  Media Type = File
}
# Generic catalog service
Catalog {
  Name = MyCatalog
  dbname = bacula; user = bacula; password = ""
}
# Reasonable message delivery -- send most everything to
#   the email address and to the console
Messages {
  Name = Standard
  mail = root@localhost = all, !skipped, !terminate
  operator = root@localhost = mount
  console = all, !skipped, !saved
}

# Default pool definition
Pool {
  Name = Default
  Pool Type = Backup
  AutoPrune = yes
  Recycle = yes
}
#
# Restricted console used by tray-monitor to get the status of the director
#
Console {
  Name = Monitor
  Password = "GN0uRo7PTUmlMbqrJ2Gr1p0fk0HQJTxwnFyE4WSST3MWZseR"
  CommandACL = status, .status
}
```

Configuration du Client/File Daemon

General

La configuration du Client (ou File Daemon) est l'une des plus simples à effectuer. En principe, vous n'aurez rien à modifier au fichier par défaut en dehors du nom du Client afin d'identifier clairement les messages d'erreur.

Pour un discours général sur les fichiers de configuration et ressources incluant les types de données reconnues par Bacula, veuillez consulter le chapitre Configuration de ce manuel. Les ressources suivantes doivent être définies :

- Client – pour définir les clients qui doivent être sauvegardés.
- Director – pour définir le nom du Director et son mot de passe.
- Messages – pour définir où les messages d'erreur et d'information doivent être envoyés.

La ressource Client

La ressource Client (ou File Daemon) définit le nom du Client (en tant que client du Director), ainsi que le port sur lequel le Client écoute les connections du Director.

Client (ou FileDaemon) Début des enregistrements du client. Il ne doit y avoir qu'une seule ressource Client dans le fichier de configuration puisqu'il définit les propriétés du programme client courrant.

Name = <name> Le nom du client qui doit être utilisé par le Director lors de la connection. Généralement, c'est une bonne idée d'utiliser un nom en relation avec la machine de faon à identifier facilement les messages d'erreur si vous avez plusieurs clients. Cette directive est requise.

Working Directory = <Directory> Cette directive spécifie le répertoire dans lequel le File Daemon peut placer ses fichiers de statuts. Il ne devrait être utilisé que par Bacula, mais il peut être partagé par d'autres daemons Bacula, pourvu que les noms de daemons définis par la directive **Name** soient uniques pour chaque daemon. Cette directive est requise.

Sur les systèmes Win32, vous pouvez, dans certaines circonstances, être amené à spécifier une lettre de disque au niveau de la spécification du répertoire. Aussi, assurez-vous que ce répertoire est bien accessible

en écriture par l'utilisateur SYSTEM, faute de quoi les restaurations peuvent échouer (le fichier bootstrap transféré au File Daemon par le Director est temporairement placé dans ce répertoire avant d'être transmis au Storage Daemon).

Pid Directory = <Directory> Cette directive spécifie le répertoire dans lequel le Director peut placer son fichier d'Id de processus. Le fichier d'Id de processus est utilisé pour stopper Bacula et prévenir l'exécution simultanée de plusieurs copies de Bacula. Cette directive est requise. L'évaluation standard du shell de **Directory** est faite lorsque le fichier de configuration est lu, de sorte que des valeurs telles que **$HOME** seront correctement substituées.

Typiquement, sur les systèmes Linux, vous utiliserez la valeur **/var/run** pour cette directive. Si vous n'installez pas Bacula dans les répertoires du système, vous pouvez utiliser le **Working Directory** tel que défini ci-dessus.

Heartbeat Interval = <time-interval> Cette directive définit un intervalle de temps. Chaque "pulsation" du Storage Daemon reue par le File Daemon est transmise au Director. De plus, si aucune pulsation n'est reue du Storage Daemon (et donc pas transmise au Director) le File Daemon envoie une pulsation à ces deux daemons afin de conserver les canaux actifs. La valeur par défaut est zéro, ce qui désactive les pulsations. Cette fonction est particulièrement utile si vous avez un routeur tels que les 3com qui ne suivent pas les standards Internet et expirent une connection valide après une courte période en dépit de l'activation de *keepalive*. Il en résulte en général un message "broken pipe error".

Si vous continuez de recevoir de messages broken pipe malgré **Heartbeat Interval**, et si vous utilisez Windows, vous devriez envisager de mettre à jour votre pilote ethernet. Il s'agit d'un problème connu des pilotes NVidia NForce 3 (4.4.2 17/05/2004). Vous pouvez aussi essayer la procédure suivante suggérée par Thomas Simmons pour les machines Win32 :

Démarrer > Panneau de configuration > connections réseau

Faites un click droit sur connection pour l'adaptateur nvidia et sélectionnez "propriétés". Sous l'onglet "Général", cliquez "Configurer...". Sous l'onglet "Avancé", désactivez l'option "Checksum Offload" et cliquez "Ok" pour sauvegarder la modification.

L'absence de communication, ou des interruptions dans les communications peuvent aussi être causées par des firewalls Linux si vous avez une règle qui limite les connections ou le trafic.

Right click the connection for the nvidia adapter and select properties. Under the General tab, click "Configure...". Under the Advanced tab set "Checksum Offload" to disabled and click OK to save the change.

Lack of communications, or communications that get interrupted can also be caused by Linux firewalls where you have a rule that throttles connections or traffic.

Maximum Concurrent Jobs = <number> Où <number> est le nombre maximum de jobs qui peuvent être exécutés simultanément. La valeur par défaut est 2, mais vous pouvez mettre une valeur plus importante. Tout contact du Director (par exemple : requête de statut, de lancement de job...) est considéré comme un job, aussi, si vous souhaitez garder la possibilité de faire une requête de statut **status** dans la console alors qu'un job est en cours, vous devez régler cette valeur supérieure à 1.

FDAddresses = <IP-address-specification> Spécifie les ports et adresses sur lesquels le Director écoute les connections de consoles Bacula. La meilleure explication est probalement un exemple :

```
FDAddresses  = { ip = {
        addr = 1.2.3.4; port = 1205; }
    ipv4 = {
        addr = 1.2.3.4; port = http; }
    ipv6 = {
        addr = 1.2.3.4;
        port = 1205;
    }
    ip = {
        addr = 1.2.3.4
        port = 1205
    }
    ip = {
        addr = 1.2.3.4
    }
    ip = {
        addr = 201:220:222::2
    }
    ip = {
        addr = bluedot.thun.net
    }
}
```

où ip, ip4, ip6, addr, et port sont tous des mots-clef. Notez que les adresses peuvent être spécifiées aussi bien en quadruplets pointés qu'en notation IPv6 double pointée ou avec des noms symboliques (seulement pour la la spécification d'ip). De même, les ports peuvent être spécifié sous forme de nombres, ou avec les valeurs mnémoniques du fichier /etc/services. Si un port n'est pas spécifié, la valeur par défaut est utilisée. Si une section ip est spécifiée, la résolution peut s'effectuer avec IPv4 et IPv6. Si ip4 est spécifié, alors seule la résolution IPv4 est permise, il en va de même avec ip6.

FDPort = <port-number> Cette directive spécifie le numéro de port sur lequel le Client est en attente de connections du Director. Le

numéro spécifié ici doit s'accorder avec le numéro FDPort spécifié dans la ressource Client du fichier de configuration du Director. La valeur par défaut est 9102.

FDAddress = <IP-Address> Cette directive est optionnelle. Si elle est spécifiée, le serveur File Daemon (serveur pour les connections du Director) est alors lié à l'adresse spécifiée **IP-Address**, qui est soit un nom de domaine, soit une adresse IP spécifiée au format quadruplet pointé. Si cet enregistrement n'est pas spécifié, le File Daemon se lie alors à toute adresse disponible (c'est le comportement par défaut).

SDConnectTimeout = <time-interval> Cette directive définit l'intervalle de temps durant lequel le File Daemon tente de se connecter au Storage Daemon. La valeur par défaut est 30 minutes Si aucune connection n'est établie durant cet intervalle, le File Daemon efface le Job.

Maximum Network Buffer Size = <bytes> Où <bytes> spécifie la taille initiale du tampon réseau à utiliser avec le File Daemon. Cette taille sera revue à la baisse si elle est trop importante jusqu'à ce qu'elle soit acceptée par le système d'exploitation. Soyez circonspect avec cette directive car si la valeur est trop importante, elle sera diminuée par pas de 512 octets jusqu'à ce qu'elle convienne au système d'exploitation, ce qui peut requérir un grand nombre d'appels systèmes. La valeur par défaut est 32 768 octets.

Voici un exemple d'une définition de ressource Client valide :

```
Client {                              # this is me
  Name = rufus-fd
  WorkingDirectory = $HOME/bacula/bin/working
  Pid Directory = $HOME/bacula/bin/working
}
```

La ressource Director

La ressource Director définit le nom et le mot de passe des Directors autorisés à contacter ce client.

Director Début des enregistrements de Director. Il peut y avoir un nombre quelconque de ressources Director dans le fichier de configuration du Client. Chacune définit un Director autorisé à contacter ce Client.

Name = <name> Le nom du Director qui peut contacter ce Client. Ce nom doit être le même que celui spécifié au niveau de la ressource Director dans le fichier de configuration du Director. Cette directive est requise.

Password = <password> Spécifie le mot de passe qui doit être fourni par le Director pour être autorisé. Ce mot de passe doit être le même que celui spécifié dans la ressource Client du fichier de configuration du Director. Cette directive est requise.

Monitor = <yes—no> Si cette directive est réglée à **no** (valeur par défaut), ce Director dispose d'un accès total à ce Client. Dans le cas contraire (**yes**), ce Director est seulement autorisé à relever le statut courant de ce Client.

Veuillez noter que si ce Director est utilisé à des fins de surveillance, nous vous recommandons fortement de mettre cette directive à **yes** pour éviter de sérieux problèmes de sécurité.

Ainsi, plusieurs Directors peuvent être autorisés à utiliser les services de ce Client. Chaque Director aura un nom différent, et, en principe, un mot de passe différent.

Voici un exemple d'une définition de ressource Director valide :

```
#
# List Directors who are permitted to contact the File daemon
#
Director {
  Name = HeadMan
  Password = very_good                # password HeadMan must supply
}
Director {
  Name = Worker
  Password = not_as_good
  Monitor = Yes
}
```

La ressource Message

Voyez le chapitre La ressource Messages de ce manuel pour plus de détails sur la ressource Messages.

Il doit y avoir au moins une ressource Messages définie dans le fichier de configuration du Client.

Un exemple de fichier de configuration de Client

Voici un exemple de fichier de configuration de Client :

```
#
```

```
# Default  Bacula File Daemon Configuration file
#
#  For Bacula release 1.35.2 (16 August 2004) -- gentoo 1.4.16
#
# There is not much to change here except perhaps to
#   set the Director's name and File daemon's name
#   to something more appropriate for your site.
#
#
# List Directors who are permitted to contact this File daemon
#
Director {
  Name = rufus-dir
  Password = "/LqPRkX++saVyQE7w7mmiFg/qxYc1kufww6FEyY/47jU"
}
#
# Restricted Director, used by tray-monitor to get the
#   status of the file daemon
#
Director {
  Name = rufus-mon
  Password = "FYpq4yyI1y562EMS35bA0J0QC0M2L3t5cZObxT3XQxgxppTn"
  Monitor = yes
}
#
# "Global" File daemon configuration specifications
#
FileDaemon {                          # this is me
  Name = rufus-fd
  WorkingDirectory = $HOME/bacula/bin/working
  Pid Directory = $HOME/bacula/bin/working
}
# Send all messages except skipped files back to Director
Messages {
  Name = Standard
  director = rufus-dir = all, !skipped
}
```

Configuration du Storage Daemon

General

Le fichier de configuration du Storage Daemon a relativement peu de définitions de resources. Cependant, en raison du nombre pléthorique de media et de systèmes, il doit être hautement paramétrable. Par conséquent, il existe un nombre assez important de directives dans la déficnition de ressource Devices qui vous permet de définir toutes les caractéristiques de votre périphérique de stockage. Heureusement, avec les matériels modernes, les valeurs par défaut sont généralement suffisantes, et très peu de directives sont réellement indispensables.

Des exemples de directives de ressources device connues pour fonctionner pour beaucoup de lecteurs de bandes communs peuvent être trouvés dans le répertoire : <bacula-src>/examples/devices. La plupart seront énumérés ici.

Pour un discussion générale concernant les fichiers de configuration de Bacula, les ressources et les types de données reconnus, veuillez consulter le chapitre Configuration de ce manuel. Les définitions de ressources Storage suivantes doivent être définies :

- Storage – Pour définir le nom du Storage Daemon.
- Director – Pour définir le nom du Director et le mot de passe permettant d'y accéder.
- Device – Pour définir les caractéristiques de votre périphérique de stockage.
- Messages – Pour définir où les messages d'erreurs et d'information doivent être expédiés.

Ressource Storage

En général, les propriétés spécifiées au niveau de la ressource Storage définissent des propriétés globales du Storage Daemon. Chaque fichier de configuration de Storage Daemon doit avoir sa propre définition de ressource Storage.

Name = <Storage-Daemon-Name> Spécifie le nom du Storage Daemon. Cette directive est requise.

Working Directory = <Répertoire> Cette directive spécifie un répertoire où le Storage Daemon peut placer ses fichiers d'état. Ce répertoire ne devrait être utilisé que par Bacula, mais peut être partagé par d'autres daemons Bacula, pourvu que les noms donnés à chaque daemon soient uniques. Cette directive est requise.

Pid Directory = <Répertoire> Cette directive spécifie un répertoire où le Storage Daemon peut déposer son fichier d'Id de processus. Ce fichier est utilisé pour stopper Bacula et prévenir l'exécution simultanée de plusieurs copies de Bacula. Les substitutions shell standard sont effectuées à la lecture du fichier de configuration, de sorte que des valeurs telles que **$HOME** seront correctement substituées.

Typiquement, sur les systèmes Linux, vous utiliserez ici **/var/run**. Si vous n'installez pas Bacula dans les répertoires système, vous pouvez utiliser le répertoire de travail **Working Directory** défini plus haut. Cette directive est requise.

Heartbeat Interval = <Période> Cette directive définit la période des pulsations émises par le Storage Daemon vers le File Daemon lorqu'il (le SD) se trouve en situation d'attente du montage d'une cartouche par l'opérateur. La valeur par défaut est zéro, ce qui désactive les pulsations. Cette fonctionnalité est particulièrement utile si vous avez un routeur (tel que les 3Com) qui ne suit pas les standards Internet et expire une connection valide après une courte durée, bien que *keepalive* soit activé. Ceci produit habituellement un message d'erreur du type *broken pipe*.

Maximum Concurrent Jobs = <nombre> Où <nombre> est nombre maximal de jobs qui peuvent être exécutés simultanément. La valeur par défaut est fixée à 10, mais vous pouvez définir une valeur plus grande. Chaque connexion depuis le Director (par exemple une requête de statut, le lancement d'un job...) est considérée comme un job, aussi, si vous voulez conserver la possibilité d'utiliser la commande **status** dans la console alors qu'un job est en cours d'exécution, vous devez utiliser une valeur strictement supérieure à 1. Pour exécuter plusieurs jobs simultanément, vous devez paramétrer plusieurs autres directives dans le fichier de configuration du Director. Selon ce que vous voulez faire, il faudra intervenir sur l'un ou l'autre paramètre, mais vous devrez presque surement régler le paramètre **Maximum Concurrent Jobs** de la ressource Storage du fichier de configuration du Director, et peut-être aussi ceux des ressources Job et Client.

SDAddresses = <Adresse IP> Précise les ports et adresses sur lesquels le Storage Daemon est à l'écoute de connections du Director. En principe, les valeurs par défaut sont suffisantes, et vous n'avez pas besoin d'utiliser cette directive. La meilleure explication du fonctionnement

de cette directive est certainement un exemple :

```
 SDAddresses  = { ip = {
        addr = 1.2.3.4; port = 1205; }
    ipv4 = {
        addr = 1.2.3.4; port = http; }
    ipv6 = {
        addr = 1.2.3.4;
        port = 1205;
    }
    ip = {
        addr = 1.2.3.4
        port = 1205
    }
    ip = {
        addr = 1.2.3.4
    }
    ip = {
        addr = 201:220:222::2
    }
    ip = {
        addr = bluedot.thun.net
    }
}
```

où "ip", "ip4", "ip6", "addr", et "port" sont des mots-clef. Notez que les adresses peuvent être spécifiées sous forme de quadruplets pointés, de nom symboliques (uniquement dans la spécification "ip") ou en notation IPv6 à double points. Le port peut quand à lui être spécifié par son numéro, ou par sa valeur mnémonique du fichier /etc/services. Si un port n'est pas spécifié, la valeur par défaut est utilisée. Si une section ip est spécifiée, la résolution peut être réalisée par ipv4 ou ipv6. En revanche, si ip4 ou ip6 est spécifiée, seule la résolution correspondante fonctionne.

Vous pouvez, avec ces directives, remplacer les valeurs des directives SDPort et SDAddress montrées ci-dessous.

SDPort = <Numéro de port> Spécifie le numéro de port sur lequel le Storage Daemon écoute les connexions en provenance du Director. La valeur par défaut est 9103.

SDAddress = <Adresse IP> Cette directive est optionnelle. Lorsqu'elle est spécifiée, le Storage Daemon n'accepte de connections (de Director(s) ou de File(s) Daemon(s)) que de l'adresse spécifiée **Adresse-IP**, qui peut être soit un nom de domaine, soit une adresse IP au format quadruplet pointé. Si cette directive n'est pas spécifiée, le Storage Daemon acceptera des connections de de toute adresse valide.

Voici une définition typique d'une ressource Storage Daemon :

```
#
```

```
# "Global" Storage daemon configuration specifications appear
# under the Storage resource.
#
Storage {
  Name = "Storage daemon"
  Address = localhost
  WorkingDirectory = "~/bacula/working"
  Pid    Directory = "~/bacula/working"
}
```

La ressource Director

La ressource Director spécifie le nom du Director qui est autorisé à utiliser les services du Storage Daemon. Il peut exister plusieurs ressources Director. Le nom et le mot de passe du Director doivent s'accorder avec leurs homologues dans le fichier de configuration du Storage Daemon.

Name = <Nom-du-Director> Spécifie le nom du Director autorisé à se connecter au Storage Daemon. Cette directive est requise.

Password = <Mot-de-passe-du-Director> Spécifie le mot de passe qui doit être soumis par le Director susnommé. Cette directive est requise.

Monitor = <yes—no> Si cette directive est désactivée (**no**), ce qui est le cas par défaut, ce Director dispose d'un accès illimité à ce Storage Daemon. Dans le cas contraire, ce Director est bridé de faon à pouvoir seulement récupérer le statut courant de ce Storage Daemon.

Si ce Director est utilisé par un superviseur, nous vous recommandons fortement d'activer cette directive pour éviter de sérieux problèmes de sécurité.

Voici un exemple d'une définition de ressource Director valide :

```
Director {
  Name = MainDirector
  Password = my_secret_password
}
```

La Ressource Device

La ressource Device spécifie les détails de chaque périphérique (en général, un lecteur de bandes) qui peut être utilisé par le Storage Daemon. Un Storage Daemon peut disposer de plusieurs ressources Device. En général, les propriétés spécifiées dans la ressource Device sont spécifiques au périphérique.

Name = *Nom-de-périphérique* Spécifie le nom que le Director devra utiliser pour désigner ce périphérique. Il s'agit d'un nom logique, c'est une chaîne qui peut comporter jusqu'à 127 caractères. C'est en général une bonne idée d'utiliser un nom qui corresponde au nom "humain" du périphérique (NDT: la vo dit "the english name"). Le nom physique du périphérique est spécifié au niveau de la directive **Archive Device** décrite ci-dessous. Le nom que vous spécifiez ici est aussi utilisé dans le fichier de configuration de votre Director au niveau de la directive Device de sa ressource Storage.

Archive Device = *name-string* The specified **name-string** gives the system file name of the storage device managed by this storage daemon. This will usually be the device file name of a removable storage device (tape drive), for example "**/dev/nst0**" or "**/dev/rmt/0mbn**". For a DVD-writer, it will be for example **/dev/hdc**. It may also be a directory name if you are archiving to disk storage. In this case, you must supply the full absolute path to the directory. When specifying a tape device, it is preferable that the "non-rewind" variant of the device file name be given. In addition, on systems such as Sun, which have multiple tape access methods, you must be sure to specify to use Berkeley I/O conventions with the device. The **b** in the Solaris (Sun) archive specification **/dev/rmt/0mbn** is what is needed in this case. Bacula does not support SysV tape drive behavior.

As noted above, normally the Archive Device is the name of a tape drive, but you may also specify an absolute path to an existing directory. If the Device is a directory Bacula will write to file storage in the specified directory, and the filename used will be the Volume name as specified in the Catalog. If you want to write into more than one directory (i.e. to spread the load to different disk drives), you will need to define two Device resources, each containing an Archive Device with a different directory.

In addition to a tape device name or a directory name, Bacula will accept the name of a FIFO. A FIFO is a special kind of file that connects two programs via kernel memory. If a FIFO device is specified for a backup operation, you must have a program that reads what Bacula writes into the FIFO. When the Storage daemon starts the job, it will wait for **MaximumOpenWait** seconds for the read program to start reading, and then time it out and terminate the job. As a consequence, it is best to start the read program at the beginning of the job perhaps with the **RunBeforeJob** directive. For this kind of device, you never want to specify **AlwaysOpen**, because you want the Storage daemon to open it only when a job starts, so you must explicitly set it to **No**. Since a FIFO is a one way device, Bacula will not attempt to read a label of a FIFO device, but will simply write

on it. To create a FIFO Volume in the catalog, use the **add** command rather than then **label** command to avoid attempting to write a label. During a restore operation, if the Archive Device is a FIFO, Bacula will attempt to read from the FIFO, so you must have an external program that writes into the FIFO. Bacula will wait **MaximumOpenWait** seconds for the program to begin writing and will then time it out and terminate the job. As noted above, you may use the **RunBeforeJob** to start the writer program at the beginning of the job.

The Archive Device directive is required.

Device Type = *type-specification* The Device Type specification allows you to explicitly tell Bacula what kind of device you are defining. It the *type-specification* may be one of the following:

File Tells Bacula that the device is a file. It may either be a file defined on fixed medium or a removable filesystem such as USB. All files must be random access devices.

Tape The device is a tape device and thus is sequential access. Tape devices are controlled using ioctl() calls.

Fifo The device is a first-in-first out sequential access read-only or write-only device.

DVD The device is a DVD. DVDs are sequential access for writing, but random access for reading.

The Device Type directive is not required, and if not specified, Bacula will attempt to guess what kind of device has been specified using the Archive Device specification supplied. There are several advantages to explicitly specifying the Device Type. First, on some systems, block and character devices have the same type, which means that on those systems, Bacula is unlikely to be able to correctly guess that a device is a DVD. Secondly, if you explicitly specify the Device Type, the mount point need not be defined until the device is opened. This is the case with most removable devices such as USB that are mounted by the HAL daemon. If the Device Type is not explicitly specified, then the mount point must exist when the Storage daemon starts.

This directive was implemented in Bacula version 1.38.6.

Media Type = *name-string* The specified **name-string** names the type of media supported by this device, for example, "DLT7000". Media type names are arbitrary in that you set them to anything you want, but they must be known to the volume database to keep track of which storage daemons can read which volumes. In general, each different storage type should have a unique Media Type associated with it. The same **name-string** must appear in the appropriate Storage resource definition in the Director's configuration file.

Even though the names you assign are arbitrary (i.e. you choose the name you want), you should take care in specifying them because

the Media Type is used to determine which storage device Bacula will select during restore. Thus you should probably use the same Media Type specification for all drives where the Media can be freely interchanged. This is not generally an issue if you have a single Storage daemon, but it is with multiple Storage daemons, especially if they have incompatible media.

For example, if you specify a Media Type of "DDS-4" then during the restore, Bacula will be able to choose any Storage Daemon that handles "DDS-4". If you have an autochanger, you might want to name the Media Type in a way that is unique to the autochanger, unless you wish to possibly use the Volumes in other drives. You should also ensure to have unique Media Type names if the Media is not compatible between drives. This specification is required for all devices.

In addition, if you are using disk storage, each Device resource will generally have a different mount point or directory. In order for Bacula to select the correct Device resource, each one must have a unique Media Type.

Autochanger = *Yes—No* If **Yes**, this device belongs to an automatic tape changer, and you should also specify a **Changer Device** as well as a **Changer Command**. If **No** (default), the volume must be manually changed. You should also have an identical directive to the Storage resource in the Director's configuration file so that when labeling tapes you are prompted for the slot.

Changer Device = *name-string* The specified **name-string** must be the **generic SCSI** device name of the autochanger that corresponds to the normal read/write **Archive Device** specified in the Device resource. This gemeric SCSI device name should be specified if you have an autochanger or if you have a standard tape drive and want to use the **Alert Command** (see below). For example, on Linux systems, for an Archive Device name of **/dev/nst0**, you would specify **/dev/sg0** for the Changer Device name. Depending on your exact configuration, and the number of autochangers or the type of autochanger, what you specify here can vary. This directive is optional. See the Using Autochangers chapter of this manual for more details of using this and the following autochanger directives.

Changer Command = *name-string* The **name-string** specifies an external program to be called that will automatically change volumes as required by **Bacula**. Most frequently, you will specify the Bacula supplied **mtx-changer** script as follows:

```
Changer Command = "/path/mtx-changer %c %o %S %a %d"
```

and you will install the **mtx** on your system (found in the **depkgs** release). An example of this command is in the default bacula-sd.conf

file. For more details on the substitution characters that may be specified to configure your autochanger please see the Autochangers chapter of this manual. For FreeBSD users, you might want to see one of the several **chio** scripts in **examples/autochangers**.

Alert Command = *name-string* The **name-string** specifies an external program to be called at the completion of each Job after the device is released. The purpose of this command is to check for Tape Alerts, which are present when something is wrong with your tape drive (at least for most modern tape drives). The same substitution characters that may be specified in the Changer Command may also be used in this string. For more information, please see the Autochangers chapter of this manual.

Note, it is not necessary to have an autochanger to use this command. The example below uses the **tapeinfo** program that comes with the **mtx** package, but it can be used on any tape drive. However, you will need to specify a **Changer Device** directive in your Device resource (see above) so that the generic SCSI device name can be edited into the command (with the %c).

An example of the use of this command to print Tape Alerts in the Job report is:

```
Alert Command = "sh -c 'tapeinfo -f %c | grep TapeAlert'"
```

and an example output when there is a problem could be:

```
bacula-sd  Alert: TapeAlert[32]: Interface: Problem with SCSI interface
                  between tape drive and initiator.
```

Drive Index = *number* The **Drive Index** that you specify is passed to the **mtx-changer** script and is thus passed to the **mtx** program. By default, the Drive Index is zero, so if you have only one drive in your autochanger, everything will work normally. However, if you have multiple drives, you may specify two Bacula Device resources. The first will either set Drive Index to zero, or leave it unspecified, and the second Device Resource should contain a Drive Index set to 1. This will then permit you to use two or more drives in your autochanger. However, you must ensure that Bacula does not request the same Volume on both drives at the same time. You may also need to modify the mtx-changer script to do locking so that two jobs don't attempt to use the autochanger at the same time. An example script can be found in **examples/autochangers/locking-mtx-changer**.

Autoselect = *Yes—No* If this directive is set to **yes** (default), and the Device belongs to an autochanger, then when the Autochanger is referenced by the Director, this device can automatically be selected.

If this directive is set to **no**, then the Device can only be referenced by directly using the Device name in the Director. This is useful for reserving a drive for something special such as a high priority backup or restore operations.

Maximum Changer Wait = *time* This directive specifies the maximum time in seconds for Bacula to wait for an autochanger to change the volume. If this time is exceeded, Bacula will invalidate the Volume slot number stored in the catalog and try again. If no additional changer volumes exist, Bacula will ask the operator to intervene. The default is 5 minutes.

Maximum Rewind Wait = *time* This directive specifies the maximum time in seconds for Bacula to wait for a rewind before timing out. If this time is exceeded, Bacula will cancel the job. The default is 5 minutes.

Maximum Open Wait = *time* This directive specifies the maximum time in seconds for Bacula to wait for a open before timing out. If this time is exceeded, Bacula will cancel the job. The default is 5 minutes.

Always Open = *Yes—No* If **Yes** (default), Bacula will always keep the device open unless specifically **unmounted** by the Console program. This permits Bacula to ensure that the tape drive is always available. If you set **AlwaysOpen** to **no Bacula** will only open the drive when necessary, and at the end of the Job if no other Jobs are using the drive, it will be freed. The next time Bacula wants to append to a tape on a drive that was freed, Bacula must rewind the tape and position to the end. To avoid unnecessary tape positioning and to minimize unnecessary operator intervention, it is highly recommended that **Always Open = yes**. This also ensures that the drive is available when Bacula needs it.

If you have **Always Open = yes** (recommended) and you want to use the drive for something else, simply use the **unmount** command in the Console program to release the drive. However, don't forget to remount the drive with **mount** when the drive is available or the next Bacula job will block.

For File storage, this directive is ignored. For a FIFO storage device, you must set this to **No**.

Please note that if you set this directive to **No** Bacula will release the tape drive between each job, and thus the next job will rewind the tape and position it to the end of the data. This can be a very time consuming operation.

Volume Poll Interval = *time* If the time specified on this directive is non-zero, after asking the operator to mount a new volume Bacula will periodically poll (or read) the drive at the specified interval to see

if a new volume has been mounted. If the time interval is zero (the default), no polling will occur. This directive can be useful if you want to avoid operator intervention via the console. Instead, the operator can simply remove the old volume and insert the requested one, and Bacula on the next poll will recognize the new tape and continue. Please be aware that if you set this interval too small, you may excessively wear your tape drive if the old tape remains in the drive, since Bacula will read it on each poll. This can be avoided by ejecting the tape using the **Offline On Unmount** and the **Close on Poll** directives. However, if you are using a Linux 2.6 kernel or other OSes such as FreeBSD or Solaris, the Offline On Unmount will leave the drive with no tape, and Bacula will not be able to properly open the drive and may fail the job. For more information on this problem, please see the description of Offline On Unmount in the Tape Testing chapter.

Close on Poll= *Yes—No* If **Yes**, Bacula close the device (equivalent to an unmount except no mount is required) and reopen it at each poll. Normally this is not too useful unless you have the **Offline on Unmount** directive set, in which case the drive will be taken offline preventing wear on the tape during any future polling. Once the operator inserts a new tape, Bacula will recognize the drive on the next poll and automatically continue with the backup. Please see above more more details.

Maximum Open Wait = *time* This directive specifies the maximum amount of time in seconds that Bacula will wait for a device that is busy. The default is 5 minutes. If the device cannot be obtained, the current Job will be terminated in error. Bacula will re-attempt to open the drive the next time a Job starts that needs the the drive.

Removable media = *Yes—No* If **Yes**, this device supports removable media (for example, tapes or CDs). If **No**, media cannot be removed (for example, an intermediate backup area on a hard disk).

Random access = *Yes—No* If **Yes**, the archive device is assumed to be a random access medium which supports the **lseek** (or **lseek64** if Largefile is enabled during configuration) facility.

Minimum block size = *size-in-bytes* On most modern tape drives, you will not need or wamt to specify this directive, and if you do so, it will be to make Bacula use fixed block sizes. This statement applies only to non-random access devices (e.g. tape drives). Blocks written by the storage daemon to a non-random archive device will never be smaller than the given **size-in-bytes**. The Storage daemon will attempt to efficiently fill blocks with data received from active sessions but will, if necessary, add padding to a block to achieve the required minimum size.

To force the block size to be fixed, as is the case for some non-random

access devices (tape drives), set the **Minimum block size** and the **Maximum block size** to the same value (zero included). The default is that both the minimum and maximum block size are zero and the default block size is 64,512 bytes. If you wish the block size to be fixed and different from the default, specify the same value for both **Minimum block size** and **Maximum block size**.

For example, suppose you want a fixed block size of 100K bytes, then you would specify:

```
Minimum block size = 100K
Maximum block size = 100K
```

Please note that if you specify a fixed block size as shown above, the tape drive must either be in variable block size mode, or if it is in fixed block size mode, the block size (generally defined by **mt**) **must** be identical to the size specified in Bacula – otherwise when you attempt to re-read your Volumes, you will get an error.

If you want the block size to be variable but with a 64K minimum and 200K maximum (and default as well), you would specify:

```
Minimum block size = 64K
Maximum blocksize = 200K
```

Maximum block size = *size-in-bytes* On most modern tape drives, you will not need to specify this directive. If you do so, it will most likely be to use fixed block sizes (see Minimum block size above). The Storage daemon will aways attempt to write blocks of the specified **size-in-bytes** to the archive device. As a consequence, this statement specifies both the default block size and the maximum block size. The size written never exceed the given **size-in-bytes**. If adding data to a block would cause it to exceed the given maximum size, the block will be written to the archive device, and the new data will begin a new block.

If no value is specified or zero is specified, the Storage daemon will use a default block size of 64,512 bytes (126 * 512).

Hardware End of Medium = *Yes—No* If **No**, the archive device is not required to support end of medium ioctl request, and the storage daemon will use the forward space file function to find the end of the recorded data. If **Yes**, the archive device must support the `ioctl MTEOM` call, which will position the tape to the end of the recorded data. In addition, your SCSI driver must keep track of the file number on the tape and report it back correctly by the **MTIOCGET** ioctl. Note,

some SCSI drivers will correctly forward space to the end of the recorded data, but they do not keep track of the file number. On Linux machines, the SCSI driver has a **fast-eod** option, which if set will cause the driver to lose track of the file number. You should ensure that this option is always turned off using the **mt** program.

Default setting for Hardware End of Medium is **Yes**. This function is used before appending to a tape to ensure that no previously written data is lost. We recommend if you have a non-standard or unusual tape drive that you use the **btape** program to test your drive to see whether or not it supports this function. All modern (after 1998) tape drives support this feature.

Fast Forward Space File = *Yes—No* If **No**, the archive device is not required to support keeping track of the file number (**MTIOCGET** ioctl) during forward space file. If **Yes**, the archive device must support the `ioctl MTFSF` call, which virtually all drivers support, but in addition, your SCSI driver must keep track of the file number on the tape and report it back correctly by the **MTIOCGET** ioctl. Note, some SCSI drivers will correctly forward space, but they do not keep track of the file number or more seriously, they do not report end of meduim.

Default setting for Fast Forward Space File is **Yes**.

Use MTIOCGET = *Yes—No* If **No**, the operating system is not required to support keeping track of the file number and reporting it in the (**MTIOCGET** ioctl). The default is **Yes**. If you must set this to No, Bacula will do the proper file position determination, but it is very unfortunate because it means that tape movement is very inefficient. Fortunately, this operation system deficiency seems to be the case only on a few *BSD systems. Operating systems known to work correctly are Solaris, Linux and FreeBSD.

BSF at EOM = *Yes—No* If **No**, the default, no special action is taken by Bacula with the End of Medium (end of tape) is reached because the tape will be positioned after the last EOF tape mark, and Bacula can append to the tape as desired. However, on some systems, such as FreeBSD, when Bacula reads the End of Medium (end of tape), the tape will be positioned after the second EOF tape mark (two successive EOF marks indicated End of Medium). If Bacula appends from that point, all the appended data will be lost. The solution for such systems is to specify **BSF at EOM** which causes Bacula to backspace over the second EOF mark. Determination of whether or not you need this directive is done using the **test** command in the **btape** program.

TWO EOF = *Yes—No* If **Yes**, Bacula will write two end of file marks when terminating a tape – i.e. after the last job or at the end of the medium. If **No**, the default, Bacula will only write one end of file to

terminate the tape.

Backward Space Record = *Yes—No* If *Yes*, the archive device supports the `MTBSR ioctl` to backspace records. If *No*, this call is not used and the device must be rewound and advanced forward to the desired position. Default is **Yes** for non random-access devices. This function if enabled is used at the end of a Volume after writing the end of file and any ANSI/IBM labels to determine whether or not the last block was written correctly. If you turn this function off, the test will not be done. This causes no harm as the re-read process is precautionary rather than required.

Backward Space File = *Yes—No* If *Yes*, the archive device supports the **MTBSF** and **MTBSF ioctl**s to backspace over an end of file mark and to the start of a file. If *No*, these calls are not used and the device must be rewound and advanced forward to the desired position. Default is **Yes** for non random-access devices.

Forward Space Record = *Yes—No* If *Yes*, the archive device must support the **MTFSR ioctl** to forward space over records. If **No**, data must be read in order to advance the position on the device. Default is **Yes** for non random-access devices.

Forward Space File = *Yes—No* If **Yes**, the archive device must support the `MTFSF ioctl` to forward space by file marks. If *No*, data must be read to advance the position on the device. Default is **Yes** for non random-access devices.

Offline On Unmount = *Yes—No* The default for this directive is **No**. If **Yes** the archive device must support the `MTOFFL ioctl` to rewind and take the volume offline. In this case, Bacula will issue the offline (eject) request before closing the device during the **unmount** command. If **No** Bacula will not attempt to offline the device before unmounting it. After an offline is issued, the cassette will be ejected thus **requiring operator intervention** to continue, and on some systems require an explicit load command to be issued (**mt -f /dev/xxx load**) before the system will recognize the tape. If you are using an autochanger, some devices require an offline to be issued prior to changing the volume. However, most devices do not and may get very confused.

If you are using a Linux 2.6 kernel or other OSes such as FreeBSD or Solaris, the Offline On Unmount will leave the drive with no tape, and Bacula will not be able to properly open the drive and may fail the job. For more information on this problem, please see the description of Offline On Unmount in the Tape Testing chapter.

Maximum Volume Size = *size* No more than **size** bytes will be written onto a given volume on the archive device. This directive is used mainly in testing Bacula to simulate a small Volume. It can also be useful if you wish to limit the size of a File Volume to say less than 2GB of data.

In some rare cases of really antiquated tape drives that do not properly indicate when the end of a tape is reached during writing (though I have read about such drives, I have never personally encountered one). Please note, this directive is deprecated (being phased out) in favor of the **Maximum Volume Bytes** defined in the Director's configuration file.

Maximum File Size = *size* No more than **size** bytes will be written into a given logical file on the volume. Once this size is reached, an end of file mark is written on the volume and subsequent data are written into the next file. Breaking long sequences of data blocks with file marks permits quicker positioning to the start of a given stream of data and can improve recovery from read errors on the volume. The default is one Gigabyte.

Block Positioning = *yes—no* This directive is not normally used (and has not yet been tested). It will tell Bacula not to use block positioning when it is reading tapes. This can cause Bacula to be **extremely** slow when restoring files. You might use this directive if you wrote your tapes with Bacula in variable block mode (the default), but your drive was in fixed block mode. If it then works as I hope, Bacula will be able to re-read your tapes.

Maximum Network Buffer Size = *bytes* where *bytes* specifies the initial network buffer size to use with the File daemon. This size will be adjusted down if it is too large until it is accepted by the OS. Please use care in setting this value since if it is too large, it will be trimmed by 512 bytes until the OS is happy, which may require a large number of system calls. The default value is 32,768 bytes.

The default size was chosen to be relatively large but not too big in the case that you are transmitting data over Internet. It is clear that on a high speed local network, you can increase this number and improve performance. For example, some users have found that if you use a value of 65,536 bytes they get 5-10 times the throughput. Larger values for most users don't seem to improve performance. If you are interested in improving your backup speeds, this is definitely a place to experiment. You will probably also want to make the corresponding change in each of your File daemons conf files.

Maximum Spool Size = *bytes* where the bytes specify the maximum spool size for all jobs that are running. The default is no limit.

Maximum Job Spool Size = *bytes* where the bytes specify the maximum spool size for any one job that is running. The default is no limit. This directive is implemented only in version 1.37 and later.

Spool Directory = *directory* specifies the name of the directory to be used to store the spool files for this device. This directory is also used to store temporary part files when writing to a device that requires

mount (DVD). The default is to use the working directory.

Maximum Part Size = *bytes* This is the maximum size of a volume part file. The default is no limit. This directive is implemented only in version 1.37 and later.

If the device requires mount, it is transfered to the device when this size is reached. In this case, you must take care to have enough disk space left in the spool directory.

Otherwise, it is left on the hard disk.

It is ignored for tape and FIFO devices.

Devices that require a mount (DVD)

All the directives in this section are implemented only in Bacula version 1.37 and later and hence are available in version 1.38.6.

As of version 1.39.5, the directives "Requires Mount", "Mount Point", "Mount Command", and "Unmount Command" apply to removable file-systems such as USB in addition to DVD.

Requires Mount = *Yes—No* You must set this directive to **yes** for DVD-writers, and to **no** for all other devices (tapes/files). This directive indicates if the device requires to be mounted to be read, and if it must be written in a special way. If it set, **Mount Point**, **Mount Command**, **Unmount Command** and **Write Part Command** directives must also be defined.

Mount Point = *directory* Directory where the device can be mounted.

Mount Command = *name-string* Command that must be executed to mount the device. Before the command is executed, %a is replaced with the Archive Device, and %m with the Mount Point.

Most frequently, you will define it as follows:

```
Mount Command = "/bin/mount -t iso9660 -o ro %a %m"
```

Unmount Command = *name-string* Command that must be executed to unmount the device. Before the command is executed, %a is replaced with the Archive Device, and %m with the Mount Point.

Most frequently, you will define it as follows:

```
Unmount Command = "/bin/umount %m"
```

Write Part Command = *name-string* Command that must be executed to write a part to the device. Before the command is executed, %a is replaced with the Archive Device, %m with the Mount Point, %e is