rsnapshot HOWTO

Nathan Rosenquist <nathan@rsnapshot.org>

2004-01-20

Revision History

Revision 0.9.5	2004-07-10	NR
Relicensed document under GPL, instead of FDL		
Revision 0.9.4	2004-07-02	NR
Added description of proper crontab time settings		
Revision 0.9.3	2004-06-11	NR
Misc. updates		
Revision 0.9.2	2004-05-16	NR
Updated --link-dest info		
Revision 0.9.1	2004-01-20	NR
Added --link-dest info		
Revision 0.9	2004-01-10	NR
First draft		

rsnapshot est un outil de sauvegarde basé sur rsync. Grâce à rsnapshot, il est possible de prendre des instantanés de vos systèmes de fichiers à différents instants. Utilisant les liens durs, rsnapshot créé l'illusion de l'existance de multiples sauvegardes complétes, tout en n'occupant que la place d'une seule plus les différences. Couplé avec ssh, il est également possible de prendre des instantanés de systèmes de fichiers distants. Ce document est un tutoriel d'installation et de configuration de rsnapshot.

Table of Contents

1. Introduction

rsnapshot est un outil de sauvegarde basé sur rsync. Grâce à rsnapshot, il est possible de prendre des instantanés de vos systèmes de fichiers à différents instants. Utilisant les liens durs, rsnapshot créé l'illusion de l'existance de multiples sauvegardes complétes, tout en n'occupant que la place d'une seule plus les différences. Couplé avec ssh, il est également possible de prendre des instantanés de systèmes de fichiers distants. Ce document est un tutoriel d'installation et de configuration de rsnapshot.

rsnaphost est écrit en Perl et dépends de rsync. OpenSSH, GNU cp et le programme BSD logger sont également recommandés mais pas necessaires. Tous ceux-ci devraient être présents sur la plupart des systèmes Linux. rsnapshot est écrit avec le plus petit dénomitateur commun en tête. Au minimum il ne requiert que Perl 5.0004 et rsync. Ceci fait qu'il fonctionne sur quasiment n'importe quel système UNIX vous voudrez l'installer. Il a été testé avec succès avec Perl 5.0004 à 5.8.2, sur Debian, Redhat, Fedora, Solaris, Mac OS X, FreeBSD, OpenBSD, et IRIX.

La dernière version du programme et de ce document peuvent toujours être trouvés à l'adresse http://www.rsnapshot.org/.

1.1. Ce dont vous aurez besoin

Au minimum :*perl, rsync*

Eventuellement :*ssh, logger, GNU cp*

De plus, de bonnes compétences d'administration système aideront.

1.2. Copyright et License

Ce document, rsnapshot HOWTO, est copyrighté (c) 2004 par Nathan Rosenquist. Vous pouvez le redistribuer et/ou le modifier selon les termes de la licence GNU General Public License publiée par la Free Software Foundation ; soit selon la version 2 de la license, ou (au choix) n'importe quelle version suivante. Une copie de la license est disponible à l'adresse http://www.gnu.org/copyleft/gpl.html.

1.3. Avertissement

Aucune résponsabilité sur le contenu de ce document n'est prise. Utilisez ces concepts, exemples et informations à vos risques et périls. Il peut y avoir des erreurs et inexactitudes qui pourraient endommager votre système. Procédez avec attention, et bien que ce soit fortement peu probable, l'auteur ne prends aucune responsabilités.

All copyrights are held by their by their respective owners, unless specifically noted otherwise. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark. Naming of particular products or brands should not be seen as endorsements.

1.4. Retours et critiques

Retours et critiques sont les bienvenues à propos de ce document. Envoyez vos ajouts, commentaires et critiques à l'adresse email suivante :<nathan@rsnapshot.org>.

2. Motivation

A l'origine j'utilisait les scripts shell de Mike Rubel pour faire des instantanés rsync. Ils fonctionnaient

très bien mais il restait plusieurs choses que je voulais améliorer. J'ai du écrire deux scripts shell personnalisés pour mon serveur. Si je voulais changer le nombre d'instantanés stockés, ou les parties du systèmes de fichiers archivées, il me fallait éditer manuellement ces scripts shell. De plus, je faisait toutes les sauvegardes en local, sur une seule et même machine, sur un seul et même disque dur. Malgré ça, j'ai continué avec ce système pendant un moment et ca m'a bien servi.

Quelques mois plus tard, le controleur IDE de mon serveur web planta horiblement (lorsque j'ai tappé / **sbin/shutdown**, il me dit que la commande était introuvable). Je fut alors confronté à ce que j'ai toujours su : je n'avais jamais fait de sauvegardes distantes régulières de mon serveur et les sauvegardes locales ne m'étaient d'aucune utilité maintenant que le disque était corrompu. La principale raison est que les sauvegardes distantes automatiques n'étaient ni automatisées, ni sans effort. Evidemment, ce n'était la faute de personne à part la mienne mais ma frustration était assez grande pour que je décide d'écrire un outil qui rendrait les sauvegardes distantes automatisées tellement simples que je n'aurai plus jamais à m'en soucier. Ce but a été atteinds depuis longtemps mais le travail sur rsnapshot continue au fil des patches envoyés par des utilisateurs, des demandes de fonctionnalités et que sont trouvées des manières d'améliorer le programme.

3. Installation

Cette section vous guideras tout au long de l'installation de rsnapshot, étape par étape. Ce n'est pas la seule manière de le faire mais c'est une manière qui fonctionne et qui est bien documentée. Imporivez si vous savez ce que vous faites.

3.1. Version 30 secondes (pour les impatients)

```
./configure --sysconfdir=/etc
su
make install
cp /etc/rsnapshot.conf.default /etc/rsnapshot.conf
```

Le reste de cette section est la version longue.

3.2. Décompresser les sources

```
tar xzvf rsnapshot-1.1.15.tar.gz
```

Si vous n'avez pas GNU `tar`, vous pourriez avoir besion de faire ça en deux étapes.

```
gunzip rsnapshot-1.1.15.tar.gz
tar xvf rsnapshot-1.1.15.tar
```

3.3. Rentrer dans le dossier des sources

```
cd rsnapshot-1.1.15/
```

3.4. Décider où vous voulez installer

Par défaut, la procédure d'installation installera les fichiers dans `/usr/local`. Pour ce tutoriel, ce sera

OK mis à part que nous installerons le fichier de configuration dans `/etc`.

Nous partons du principe que `rsync`, `ssh`, and `logger` sont tous dans votre $PATH. Si ce n'est pas le cas, vous pouvez spécifier le chemin de chacun de ces programmes en utilisant l'habituel Autoconf *--with-program=/path/to/program* syntaxe. Par exemple, si Perl était dans `/opt/bin/perl` et si rsync était dans `/home/me/bin/rsync`, vous pourriez lancer configure comme ceci :

```
./configure --with-perl=/opt/bin/perl --with-rsync=/home/me/bin/rsync
```

3.5. Lancer le script de configuration

Ce script interrogera et analysera votre système pour savoir ou se trouvent les différents programmes externes dont rsnapshot a besoin. Il générera également le Makefile qui servira à installer le programme. Le script de de configuration accept des arguments qui peuvent servir à lui dire où installer le programme, mais aussi où trouver les autres programmes necessaires. Pour cette installation, la seule option différente des options par défaut que nous voulons spécifier est que nous voulons placer le fichier de configuration dans `/etc`. Tappez cette commande dans un shell :

```
./configure --sysconfdir=/etc
```

Si tout se passe bien, vous êtes pret a installer le programme. S'il y a eu un problème, il devrait être descriptif. Le plus souvent le problème viens du fait que quelque chose de necessaire n'a pas été trouvé (comme rsync ou perl). Si ca vous arrive, vous devez trouver ou se trouve le programme manquant dans votre système, ou l'installer si necessaire. Si vous savez où il se trouve mais que le script de configuration ne l'a pas trouvé, vous pouvez spécifier le chemin en utilisant l'option décrite ci-dessus : *--with-program=/path/to/program*

3.6. Installer le programme

Si vous avez suivi les instructions jusqu'ici, vous aurez configuré rsnapshot pour qu'il soit installé dans `/usr/local`, avec son fichier de configuration dans `/etc`. dans ces conditions, il est necessaire de devenir root pour installer le programme. Il est maintenant tant de le faire. Bien sûr vous aurez besoin du mot de passe de root pour le faire :

```
su
```

Cette commande vous demandera le mot de passe. Si vous avez les privilèges root, vous le saviez déjà :)

Maintenant, pour installer rsnapshot, tappez la commande suivante :

```
make install
```

Ca installera rsnapshot avec tous les paramètres que vous avez spécifiés à l'étape du ./configure. Si tout se passe bien, les fichiers suivants seront installés dans votre système :

`/usr/local/bin/rsnapshot` Le programme rsnapshot

`/usr/local/man/man1/rsnapshot.1` La page de man

`/etc/rsnapshot.conf.default` Le fichier de configuration exemple

Si, plus tard, vous décidez que vous ne voulez plus rsnapshot dans votre système, effacez simplement les

fichiers listés ci-dessus. Bien sûr, si vous avez installé avec des options différentes, l'emplacement des fichiers peut être différent.

4. Configuration

4.1. Créer le fichier de configuration

Le fichier de configuration n'est ni créé, ni installé pendant le processus d'installation, ce qui fait qu'il n'y a aucune possibilité d'écraser par accident un fichier de configuration existant pendant une mise à jour. Cependant, un exemple fonctionnel que vous pouvez copier est fourni. Pour copier le fichier de configuration exemple à l'endroit où rsnapshot cherchera le vrai fichier de configuration :

```
cp /etc/rsnapshot.conf.default /etc/rsnapshot.conf
```

Vous devriez éviter de modifier`/etc/rsnapshot.conf.default`, simplement car c'est un exemple qui fonctionne que vous pourriez avoir envie de regarder plus tard. De plus, si vous faites une mise à jour, `rsnapshot.conf.default` sera toujours mis à jour vers la dernière version, alors que votre vrai fichier de configuration sera conservé tel quel.

4.2. Où trouver plus d'information

Le fichier de configuration `rsnapshot.conf` est bien documenté et il devrait être assez explicite. Pour une référence complète des différentes options, consultez la man page de rsnapshot. Tappez :

```
man rsnapshot
```

Ca vous affichera la documentation complète. Cependant, cette man page pars du principe que vous savez déjà ce que vous voulez faire. Si vous voulez juste avoir quelque chose qui fonctionne, ce tutoriel est un meilleur guide. Si votre système ne peut trouver la man page, `/usr/local/man` n'est probablmeent pas dans votre variable d'environement $MANPATH. Ceci dépasse le champ de ce document mais si ca ne fonctionne pas pour vous, vous pouvez toujours lire la dernière man page sur le site web de rsnapshot à l'adresse http://www.rsnapshot.org/

4.3. Modifier le fichier de configuration

Dans cet exemple, nous utiliseront le dossier `/.snapshots/` pour contenir les instantannés du système de fichiers. On l'appelera la : “snapshot root”. Vous pouvez la placer où vous voulez, à un endroit ou vous avez beaucoup d'espace disque. Cependant, les exemples de ce document partent du principe que vous n'avez pas changé ce paramètre, vous devrez donc le substituer dans votre esprit si vous devez l'utilisez plus tard.

De plus, veuillez noter que les champs sont séparés par des tabulations, pas des espaces. Ceci permet de spécifier des chemins contenant des esapces.

4.3.1. cmd_cp

Si activé, le paramètre *cmd_cp* doit contenir le chemin vers le programme GNU `cp` sur votre système de fichiers. Si vous utilisez Linux, décommentez ceci en retirant le diese (#) en début de ligne. Si vous utilisez BSD, Solaris, IRIX ou d'autres variantes d'UNIX, vous devriez le laisser commenté.

Ce qui rend GNU `cp` si special c'est que contrairement au traditionnel `cp` UNIX, c'est qu'il a la

possibilité de faire des “copies” récursives de repertoires en tant que liens durs. Si vous n'avez pas GNU `cp`, il existe une routine dans rsnapshot qui équivaut à peut près à cette fonctionnalité (bien qu'il ne supporte pas les fichiers ésotériques commes les noeuds de périphériques, les FIFOs, les sockets etc...).

Si vous avez une version supérieure ou égale à 2.5.7 de `rsync`, le paramètre link_dest de rsnapshot remplace le paramètre *cmd_cp*. *link_dest* est la seule manière de supporter tous types de fichiers sur toutes platte-forme.

4.3.2. cmd_rsync

Le paramètre *cmd_rsync* ne doit pas être commenté et doit pointer vers une version fonctionnelle de `rsync`. S'il ne le fait pas, le programme ne fonctionnera tout simplement pas. Veuillez noter égalment que si vous utilisez IRIX, il existe un autre programme nommé `rsync` différent du “vrai” `rsync` que la plupart des gens connaissent. Si vous êtes sur une machine IRIX, vous devrier vérifier ce point.

4.3.3. cmd_ssh

Si vous avez `ssh` d'installé sur votre système vous voudrez décommenter le paramètre *cmd_ssh*. Une fois de plus, ceci est fait en enlevant le diese en face du paramètre. En activant `ssh`, vous pouvez faire des instantanés depuis plusieurs systèmes distants. Si vous n'avez pas `ssh`, ou si vous pensez ne prendre des instantanés que du système de fichiers local, vous pouvez sans problème laisser ce paramètre commenté.

4.3.4. cmd_logger

Le paramète *cmd_logger* spécifie le chemin vers le programme `logger`. `logger` est une interface en ligne de commande vers syslog. Lisez la page de man de `logger` pour plus de détails. `logger` doit faire partie de la base de la plupart des systèmes UNIXes. Il n'a pas changé depuis 1993 environ. C'est bon pour la stabilité cross-platform :) Si vous commentez ce paramètre, ca désactivera le support syslog dans rsnapshot. Il est recommendé le laisser activé.

4.3.5. link_dest

Si vous avez une version supérieur ou égale à 2.5.7 de `rsync`, vous devriez vouloir l'activer. Avec *link_dest* activé, rsnapshot utilise rsync pour créer les liens durs recursifs, plutot que GNU cp. Avec *link_dest* activé, tout type de fichier sur votre système peut être sauvegardé, sur n'importe quel système d'exploitation. Pour tirer le meilleur de rsnapshot, *link_dest* devrait être activé. Soyez averti, cependant, que si un hôte distant n'est pas accessible pendant une sauvegarde, ca videra le contenu du dernier instantané pour ce point de sauvegarde sur le serveur rsnapshot. Un re-sync complet sera necessaire la prochaine fois que l'hote deviens disponible. Nous espérons corriger ceci dans le futur.

Sans cette fonctionnalité (ou GNU cp activé sur Linux), certains types de fichiers spéciaux, comme les FIFOs, les sockets et les pipes ne seront pas sauvegardés. Si vous pensez utiliser rsnapshot pour faire une sauvegarde complète de votre système, assurez vous d'activer *link_dest*, ou au minimum d'activer GNU cp sur un système Linux.

4.3.6. interval

rsnapshot n'a aucune idée de la fréquence des instantanés que vous désirez. Le modéle de sauvegarde de chacun est différent. Pour spécifier combien de données sauvegarder, vous devez dire à rsnapshot quelles “intervalles” conserver et combien de chaque. Une intervalle, dans le contexte du fichier de configuration de rsnapshot, est une unité de mesure du temps. Elles peuvent être nommées comme vous le voulez (tant que ce nom reste alphanumérique). Par conventions nous appelerons les notres *hourly* et *daily*. Dans cet exemple nous voulons prendre un instantanné toutes les 4 heures, ou six fois par jour (c'est les intervalles *hourly*). Nous voulons également conserver un deuxième jeu, pris une fois par jour et stocké pour une semaine (ou sept jours). C'est la configuration par défaut comme vous pouvez le voir dans le fichier de configuration :

```
interval    hourly  6
interval    daily   7
```

Il y a quelques autres entrées mais vous pouvez les ignorer pour l'instant ou les commenter.

Veuillez noter que l'intervale *hourly* est spécifiée en premier. C'est très important. La première ligne *interval*est considérée comme étant la plus petite unité de temps. Chaque ligne supplémentaire désignant une intervalle plus grande. Donc, si vous ajoutez une intervalle *yearly*, elle doit aller en bas de la liste. Et si vous ajoutez une intervale *minutes*, elle doit aller avant hourly. Il faut notre également que les instantanés sont “créés” depuis la plus petite intervalle vers la plus grande. Dans cet exemple, les instantanés journaliers sont créés depuis le plus vieux instantané horaire, pas depuis le système de fichier à sauvegarder.

4.3.7. backup

C'est dans cette section que vous dites à rsnapshot quels fichiers vous voulez voir sauvegardés. Vous mettez un paramètre "backup" en premier, suivi par le chemin complet vers le dossier ou le chemin réseau vous voulez sauvegarder. La troisième colonne contient le chemin relatif à la snapshot root où vous voulez placer les backups. Regardons un exemple :

```
backup      /etc/       localhost/etc/
```

Dans cet exemple, *backup* nous informe que c'est un point de sauvegarde. `/etc/` est le chemin complet vers le repertoire que nous voulons sauvegarder, et `localhost/etc/` est un dossier au sein de *snapshot_root* où placer les sauvegardes. Utiliser *localhost* comme dossier de destination n'est qu'une convention. Vous pourriez très bien utiliser `etc/`, ou le vrai nom de domaine du serveur à la place de *localhost*. Si vous prenez des instantanés de plusieurs machines sur un serveur de sauvegarde dédié, c'est une bonne idée d'utiliser leurs noms de domaine respectifs comme nom de dossier afin de savoir quels fichiers viennent de quel serveur.

En plus des chemins complet vers le système de fichier local, vous pouvez aussi sauvegarder dse systèmes distant en utilisant `rsync` dans `ssh`. Si vous avez installé et activé `ssh` (via le paramètre *cmd_ssh*), vous pouvez specifier un chemin comme ceci :

```
backup      root@example.com:/etc/     example.com/etc/
```

Le comportement est fondamentalement le même mais vous devez prends quelques choses en plus en compte.

- Le serveur ssh doit tourner sur example.com
- Vous devez avoir accès au compte que vous specifiez sur la machine distante, dans ce cas, l'utilisateur root sur example.com.
- Vous devez avoir un accès basé sur des clefs pour l'utilisateur root sur example.com, sans passphrase. Si vous voulez faire les sauvegarde en tant qu'un autre utilisateur, vous pouvez le spécifier à la place de root pour la source (ex: user@domain.com). Veuillez noter qu'autoriser les connexions sans passphrases amène un risque qui peut ne pas être acceptable dans votre situation. Asurez-vous de sécuriser correctement le serveur de backup ! Pour plus d'information sur comment configurer tout ça, veuillez consulter la page de man de ssh, ou un tutoriel sur comment utiliser les clefs publiques et privées avec ssh. Vous trouverez que les connections par clefs sont meilleures en de nombreux points, pas seulement pour rsnapshot mais par praticité et sécurité. Ce que vous pouvez faire pour diminuer les domages potentiels d'une faille sur un serveur de sauvegarde est de créer des utilisateurs supplémentaires sur les machines clientes avec uid et gid mis à 0, mais avec un shell comme scponly.

- Cette sauvegarde s'effectue à travers le réseau, elle peut donc être plus lente. Vu qu'elle utilise `rsync`, c'est surtout visible pendant la première sauvegarde. Suivant quelle quantité de vos données changent, les sauvegardes suivantes devraient aller beaucoup, beaucoup plus vite vu que *rsync* n'envoie que les différences entre les fichiers.

4.3.8. backup_script

Avec ce paramètre, la deuxième colonne contient le chemin complet vers un script de sauvegarde executable, et la troisième contient le chemin local ou vous voulez stocker (tout comme avec le paramètre "backup"). Par exemple :

```
backup_script      /usr/local/bin/backup_pgsql.sh       localhost/postgres/
```

Dans cet exemple, rsnapshot lancera le script `/usr/local/bin/backup_pgsql.sh` dans un dossier temporaire, puis synchronisera le restultat dans le dossier `localhost/postgres/` dans la snapshot root. Vous pouvez trouver le script exemple backup_pgsql.sh dans le dossier `utils/` de la distribution source. Vous pouvez le modifier pour votre système.

Votre script doit simplement écrire tout le contenu ou quoi qu'il crée dans le repertoire d'où il est lancé. Il peut créer autant de fichiers et/ou dossiers que necessaire, mais ne pas en placer dans un chemin prédeterminé. rsnapshot créé un dossier temporaire, va dans ce dossier, execute le script de sauvegarde puis synchronise le contenu de ce dossier temporaire vers le chemin local que vous avez spécifié dans la troisième colonne. Un script typique pourrait archiver le contenu du base de donnée. Il ressemblerai à ça :

```
#!/bin/sh

/usr/bin/mysqldump -uroot mydatabase > mydatabase.sql
/bin/chown 644 mydatabase.sql
```

Il y a quelques scripts d'exemple dans le dossier `utils/` de la distribution source de rsnapshot pour vous donner plus d'idées.

Souvenez-vous que ces scripts de sauvegarde seront applés avec l'utilisateur qui fait tourner rsnapshot. Dans notre exemple, c'est root. Assurez-vous que vos script de sauvegarde appartiennent à root, et qu'ils ne sont pas modifiables par qui que ce soit d'autre. Si vous ne faites pas ça, quiconque qui aurait accès en écriture à ces scripts sera à même d'y ajouter des commandes qui seront executées par l'utilisateur root. S'ils sont malicieux, ils pouraient prendre possiession de votre serveur.

4.4. Tester votre fichier de configuration

Une fois tous vox changements effectués, vous devriez vérifier que le fichier de configuration est syntaxiquement valide et que tous les programmes necessaires sont bien où vous pensez qu'ils sont. Pour ceci, lancez rsnapshot avec l'argument configtest :

```
rsnapshot configtest
```

Si tout est bon, il devrait dire If all is well, it should say `Syntax OK`. S'il y a un problème il devrait vous dire exactement ou il est. Assurez-vous que le ficiher de configuration utilise dse tabulations et pas des espaces etc...

L'étape finale pour tester votre configuration est de lancer rsnapshot en mode test. Il affichera la liste des opérations qu'il effectuera, sans les effectuer vraiment. Pour faire ce test, lancez cette commande :

```
rsnapshot -t hourly
```

Cette commande dit à rsnapshot de simuler une sauvegarde horaire. Il devrait afficher les commandse qu'il effectuerai s'il s'executait pour de vrai. Veuillez-notre que la sortie d'un test peut être légèrement différente de l'éxecution réelle, simplement parce que le test de fait pas les choses qui devront être vérifiées plus tard dans le programme (ex. si le programme créera un dossier en testera ensuite son existance , le test il dira qu'il essayera de créer un dossier qui existe déjà lors de l'execution réelle du programme). Encore une fois, ces différences ne devraient être que minime, ne vous en souciez pas.

5. Automatisation

Maintenant que votre fichier de configuration est fait, il est temps de faire en sorte que rsnapshot soit lancé par cron. En tant que l'utilisateur root, editez le crontab de root en tappant :

```
crontab -e
```

Vous pourriez créer un fichier crontab, le conserver et le charger ici, mais les concepts sont les mêmes. Entre les informations suivantes dans le crontab de root :

```
0 */4 * * *       /usr/local/bin/rsnapshot hourly
30 23 * * *       /usr/local/bin/rsnapshot daily
```

En règle générale, c'est une bonne idée de programmer les intervalles les plus larges un peu avant les moins larges. Par exemple, dans le crontab ci-dessus, notez que *daily* s'execute 30 minutes avant *hourly*. Ca permet d'éviter des interbloquages qui pourraient survenir si *daily* essaye de s'executer avant que *hourly* ait terminé. La même stratégie devrait être utilisée afin que l'intervalle *weekly* s'execute avant *daily* et ainsi de suite.

6. Mode de fonctionnement

Toutes les sauvegardes sont stockées dans la snapshot root. Par défaut, c'est le dossier /.snapshots/. Au sein de ce dossier, d'autres dossiers sont crées pour les différentes intervalles spécifiées. Au départ il est vide mais apres une semaine d'execution de rsnapshot, il devrait ressembler à ça :

```
[root@localhost]# ls -l /.snapshots/
drwxr-xr-x    7 root     root         4096 Dec 28 00:00 daily.0
drwxr-xr-x    7 root     root         4096 Dec 27 00:00 daily.1
drwxr-xr-x    7 root     root         4096 Dec 26 00:00 daily.2
drwxr-xr-x    7 root     root         4096 Dec 25 00:00 daily.3
drwxr-xr-x    7 root     root         4096 Dec 24 00:00 daily.4
drwxr-xr-x    7 root     root         4096 Dec 23 00:00 daily.5
drwxr-xr-x    7 root     root         4096 Dec 22 00:00 daily.6
drwxr-xr-x    7 root     root         4096 Dec 29 00:00 hourly.0
drwxr-xr-x    7 root     root         4096 Dec 28 20:00 hourly.1
drwxr-xr-x    7 root     root         4096 Dec 28 16:00 hourly.2
drwxr-xr-x    7 root     root         4096 Dec 28 12:00 hourly.3
drwxr-xr-x    7 root     root         4096 Dec 28 08:00 hourly.4
drwxr-xr-x    7 root     root         4096 Dec 28 04:00 hourly.5
```

A l'intérieur de chacun de ces dossiers il y a une sauvegarde “complète”. Les dossiers que vous avez spécifiés avec les paramètres *backup* et *backup_script* se retrouvent dirrectement dans ces dossiers. Dans

notre exemple :

```
backup          /etc/           localhost/etc/
```

Le dossier `/etc/` est initiallement sauvegardée dans `/.snapshots/hourly.0/localhost/etc/`

Chaque fois suivante où rsnapshot est exécuté avec la commande *hourly*, une rotation sera effectuée sur les dossiers `hourly.X`, ensuite le dossier `hourly.0` est “copié” (en utilisant les liens durs) dans `hourly.1`

Quand **rsnapshot daily** est exécuté, une rotation est effecuée sur les dossiers `daily.X`, puis le contenu de `hourly.5` est copié dans `daily.0`.

`hourly.0` contiendra toujours l'instantané le plus récent, et `daily.6` contiendra toujours un instantané datant d'une semaine. Sauf si les fichiers changent entre les instantanés, les sauvegardes “complètes” ne sont réellement que de multiples liens durs vers les mêmes fichiers. Par exemple, si votre `/etc/passwd` ne change pas durant une semaine `hourly.0/localhost/etc/passwd` et `daily.6/localhost/etc/passwd` contiendront, literallement, exactement le même fichier. C'est ce qui fait que rsnapshot est si efficace du point de vue de l'espace disque. Si à un moment le fichier change, la sauvegarde suivante déliera le lien dur dans `hourly.0` et le remplacemera par un nouveau fichier. Ce qui prendra deux fois plus de place qu'à la sauvegarde précédente. Mais cette méthode prends considérablement moins de place que si les copies complétent l'étaient réellement et que ce ficiher était stocké 13 fois.

Souvenez-vous que si vous utilisez des intervalles différentes de celles prise en exemple, la première listée est celle qui se voit mise à jour directement depuis le système de fichiers. Toutes les sauvegardes suivantes piochent dans les intervalles précédentes. Par exemple, si vous aviez les intervalles *weekly*, *monthly*, et *yearly* (dans cet ordre), l'intervalle weekly se verrait mise à jour directement depuis le système de fichiers, l'intervalle monthly serait mise à jour depuis weekly, et enfin, l'intervalle yearly se verraient mise à jour depuis monthly.

7. Réstaurer les sauvegardes

Lors de son premier lancement, rsnapshot créé le dossier *snapshot_root* (`/.snapshots/` par défaut). Il donne à ce dossier le mode de permission suivant : 700 ; ceci pour une bonne raison. Le dossier des sauvegardes contiendra surement des ficihers appartenants à différents utilisateurs de votre système. Si certains de ces fichiers étaient modifiables (certains le seront surement), les utilisateurs auraient toujours la possibilité de modifier leurs fichiers. Autrement dit, s'ils peuvent accéder directement aux sauvegardes, ils peuvent les modifier, et l'intégrité de ces sauvegardes ne peut être garantie. Quel interet de faire des sauvegardes si n'importe qui peut les modifier ?

7.1. root seulement

La solution la plus simple, la moins flexible aussi, est d'interdire l'accès à la snapshot root à tous les utilisateurs. L'utilisateur root y aura évidemment toujours accès, et comme dans tout ce qui tourne autour de l'administration système, il est considéré qu'il ne fera pas n'importe quoi. Cependant, en interdisant simplement l'accès à tout le monde, l'utilisateur root sera le seul a pouvoir créer des sauvegardes. Suivant votre situation cela peut être acceptable, ou pas. Pour une configuration simple ou une machine mono-utilisateur, vous n'aurez peut-être pas besoin de plus.

7.2. Tous les utilisateurs

Si vous voulez que tous les utilisateurs puissent créer leurs propres sauvegardes, il va vous falloir travailler un peu plus (surement moins à long terme :). Il apparaît que le meilleur moyen est de créer un dossier conteneur pour la snapshot root avec le mode de permission 700 en donnant à la snapshot root le

mode de permission 755 puis de monter cette dernière en lecture seule pour les utilisateurs. On peut faire ça en utilisant Samba et NFS. Voyons comment faire en utilisant NFS sur la même machine :

Définissez la variable snapshot_root dans le fichier `/etc/rsnapshot.conf` à `/.private/.snapshots/`

```
snapshot_root       /.private/.snapshots/
```

Créez le dossier conteneur :

```
mkdir /.private/
```

Créez la vraie snapshot root :

```
mkdir /.private/.snapshots/
```

Créez le point de montage en lecture seule de la snapshot root :

```
mkdir /.snapshots/
```

Placez les bons mode de permissions sur ces nouveaux dossiers :

```
chmod 0700 /.private/
chmod 0755 /.private/.snapshots/
chmod 0755 /.snapshots/
```

Dans le fichier `/etc/exports`, ajoutez `/.private/.snapshots/` comme export NFS en lecture seule :

```
/.private/.snapshots/  127.0.0.1(ro,no_root_squash)
```

Dans le fichier `/etc/fstab`, montez `/.private/.snapshots/` en lecture seule sur `/.snapshots/`

```
localhost:/.private/.snapshots/   /.snapshots/   nfs    ro   0 0
```

Il vous faut maintenant redemmarer votre serveur NFS.

Montez maintenant la snapshot root en lecture seule :

```
mount /.snapshots/
```

Pour tester le tout, allez dans le dossier /.snapshot/ avec l'utilisateur root. En lecture seule, même root ne devrait pas pouvoir y écrire. En tant que root, essayez :

```
touch /.snapshots/testfile
```

Cette commande devrait échouer en parlant de permissions insuffisantes. C'est exactement ce que l'on veut : les utilisateurs ne pourront pas jouer avec les instantanés.

A présent, tous ce que vos utilisateurs ont à faire pour récupérer d'anciens fichiers est d'aller dans le dossier /.snapshots/, choisir l'intervalle qu'ils veulent et naviguer dans le système de fichiers pour trouver les fichiers qu'ils recherchent. NFS les empêchera de modifier quoi que ce soit, mais ils pourront copier tout ce qu'ils pouvaient lire précédemment. Toutes les permissions de système de fichier original sont toujours présentes mais le montage NFS en lecture seule empêchera toute opération d'écriture.

Veuillez noter que certaines configuration de NFS vous interdiront l'accès aux fichiers appartenants à l'utilisateur root et les rendront lisibles uniquement par root. Dans ce cas, il vous faudra récupérer les sauvegardes de root depuis la "vraie" snapshot root, et laisser les utilisateurs sans privilèges prends les leurs depuis le montage NFS en lecture seule.

8. Conclusion

Si vous avez suivi les instructions de ce document, rsnapshot devrait être installé et configuré pour effectuer des sauvegardes automatiques de votre systèmes. Si ça ne fonctionne pas, revenez en arrière et vérifiez bien chacune des étapes afin de tenter d'isoler le problème.

La quantité d'espace disque occupée devrait égaler la taille d'une sauvegarde complète plus les copies éventuelles des fichiers qui auront changés. La création de tous les liens durs prends une petite place supplémentaire mais ce n'est pas grand chose. Sur mon système, ajouter un second intervalle correspondant à 3Go complétement identique au précédent occupa 15Mo.

La dernière version de ce document et du programme rsnapshot peuvent toujours être trouvés à l'adresse http://www.rsnapshot.org/

9. Pour plus d'information

Sites web

Les scripts originaux de Mike Rubel, sur lesquels ce projet est basé	http://www.mikerubel.org/computers/rsync_snapshots/
Perl	http://www.perl.org/
GNU cp (du paquettage coreutils)	http://www.gnu.org/software/coreutils/
rsync	http://rsync.samba.org/
OpenSSH	http://www.openssh.org/
rsnapshot	http://www.rsnapshot.org/