Administration Systèmes & Réseaux

Archivage et planification des tâches

May 3, 2015

Houcemeddine HERMASSI

houcemeddine.hermassi@enit.rnu.tn

École Nationale d'Ingénieurs de Carthage ENI-CAR
Université Carthage
Tunisie

Plan de cour

Archivage et backup

Planification des tâches

Les outils de sauvegarde

La sauvegarde est un travail important de l'administrateur puisqu'en cas de gros problème, on passe généralement par une restauration du système depuis une sauvegarde, ou une image du système lorsque celui-ci était encore intègre (bon fonctionnement, pas de corruption). Chaque Unix est fourni avec des commandes et des procédures de sauvegarde qui lui sont propres. On distingue tout de même quelques outils communs.

Commandes, plans, scripts

- Pour la sauvegarde de fichiers et d'arborescences, utilisez les commandes **tar** et **cpio**. Ces commandes sauvent une arborescence, et pas un système de fichiers. On peut faire coïncider les deux.
- Pour la sauvegarde physique de disques et de systèmes de fichiers (des dumps), utilisez la commande **dd**.

Une sauvegarde incrémentale consiste à sauvegarder une première fois la totalité des données, puis ensuite uniquement les fichiers modifiés. On trouve aussi sous forme de logiciels libres ou dans le commerce des solutions plus pointues de sauvegarde (Networker par exemple). L'administrateur aura parfois à définir des scripts de sauvegarde et de restauration adaptés au cas par cas (partition système, données applicatives...) et à automatiser quand c'est possible l'exécution de ceux-ci en fonction de la date, l'heure ou la charge de la machine.

Commandes, plans, scripts

Il sera aussi très important de définir un plan de sauvegarde, en se posant les bonnes questions :

- Que faut-il sauvegarder ?
- Avec quelle fréquence ?
- Combien de temps conservera-t-on les sauvegardes, à quel endroit, en combien d'exemplaires ?
- À quel endroit sera stocké l'historique des sauvegardes ?
- Quel est le support le plus approprié ?
- Quels sont les besoins, en capacité, du support de sauvegarde ?
- Combien de temps prévoit-on pour sauvegarder un fichier, un système de fichiers et est-ce raisonnable ?
- La sauvegarde doit-elle être automatique ou manuelle ?
- Quelle est la méthode de sauvegarde la plus appropriée ?

Chaque cas étant unique, cet ouvrage ne peut répondre à toutes ces questions. Les réponses dépendent de l'environnement cible (production, intégration, tests, etc.). Cependant envisagez toujours la mise en place d'une sauvegarde système (racine, /opt, /usr, /var, /boot, etc.) après une installation et avant une modification importante, au cas où il faudrait revenir en arrière.

Voici quelques autres commandes

- **mt** : contrôle d'une bande magnétique.
- **touch** : met la date de dernière modification à l'heure actuelle, pour forcer une sauvegarde incrémentale.
- **find** : sélectionne les fichiers à sauvegarder.
- **compress** et **uncompress** : compression et décompression des fichiers.
- **gzip**, **gunzip**, **zcat**, compression et décompression au format GnuZip.

Tar

La commande **tar** est simple et efficace. Elle crée des archives des fichiers, y compris l'arborescence de fichiers, sur tout type de support y compris dans une autre fichier (archive à l'extension .tar). L'archive ainsi créée peut s'étendre sur plusieurs volumes : quand la bande ou la disquette est pleine, c'est à l'utilisateur d'en insérer une nouvelle et la sauvegarde/restitution continue.

Archiver

La syntaxe est la suivante :

```
tar cvf nom_archive Fichier(s)"
```

Par exemple pour placer dans une archive tar le répertoire Desktop :

```
$ tar cvf desktop.tar Desktop/"
```

Les paramètres sont les suivants :

- `c` : création d' archive,
- `v` : mode bavard, tar indique ce qu'il fait,
- `f` : le paramètre suivant est le nom de l'archive.

Lister

La syntaxe est la suivante :

```
tar tvf nom_archive"
```

Pour lister le contenu de l'archive précédente :

```
$ tar tvf desktop.tar"
```

Le paramètre `t` liste le contenu de l'archive.

Restauration

Pour restaurer le contenu d'une archive la syntaxe est :

```
tar xvf nom_archive fichiers "
```

Pour restaurer l'archive précédente :

```
$tar xvf desktop.tar "
```

Le paramètre `x` permet l'extraction de l'ensemble des fichiers de l'archive, ou du ou des fichiers spécifiés à la suite du nom de l'archive.

Autres paramètres

La commande **tar** de gnu permet de gérer les formats de compression directement :

- z : l'archive est compressée au format gzip.
- Z : l'archive est compressée au format compress.
- j : l'archive est compressée au format bzip2.

Ainsi les commandes précédentes pour le format de compression gzip deviennent :

```
$ tar cvzf desktop.tar.gz Desktop/ "
```

Notez la différence de taille. Les options de compression peuvent être utilisées avec c, t et x. Notez que c'est l'archive finale qui est compressée, par les fichiers individuellement. Il peut être préférable de ne pas spécifier d'option de compression si vous sauvez sur une bande dont le lecteur gère lui-même la compression.
Si votre archive est compressée et qu'elle est à destination d'un autre système, ou que vous souhaitez garder une compatibilité avec les paramètres par défaut de tar, vous pouvez procéder comme ceci :

```
$ gzip -cd desktop.tar.gz | tar xvf -"
```

Le paramètre `-d` précise à **gzip** de décompresser le fichier, tandis que `-c` passe le résultat par la sortie standard. Le - final indique à tar de récupérer le flux par l'entrée standard.

cpio

La commande **cpio** sauvegarde sur la sortie standard les fichiers dont on saisit les noms sur l'entrée standard, par défaut l'écran et le clavier. Vous devez donc utiliser les redirections. Cpio ne compresse pas les archives. C'est à vous de le faire.

Archiver

La syntaxe générale est :

```
cpio -oL "
```

Les paramètres les plus utilisés sont :

- `-o` : output, création de la sauvegarde en sortie.
- `-L` : sauve les fichiers liés et pas les liens symboliques.
- `-v` : mode bavard « verbose », informations détaillées.
- `-c` : sauvegarde des attributs des fichiers sous forme ASCII (pour l'échange entre divers OS).

Voici comment archiver et compresser le répertoire Desktop :

```
find Desktop -print| cpio -ocv | gzip > archive.cpio.gz "
```

Archivage et backup
cpio

Lister

La syntaxe est la suivante :

```
cpio -it archive "
```

Les paramètres sont :

- `-i` : lecture de l'archive en entrée.
- `-t` : comme pour tar, liste le contenu de l'archive.

```
$ cat archive.cpio.gz | gzip -cd | cpio -it "
```

Restauration

La syntaxe générale est :

```
cpio -i[umd]"
```

- `-u` : restauration inconditionnelle, avec écrasement des fichiers qui existent déjà. Par défaut les fichiers ne sont pas restaurés si ceux présents sur le disque sont plus récents ou du même âge.
- `-m` : les fichiers restaurés conservent leur dernière date de modification.
- `-d` : cpio reconstruit l'arborescence des répertoires et sous-répertoires manquants.

Pour restaurer l'archive précédente :

```
$ cat archive.cpio.gz | gzip -cd | cpio -iuvd "
```

Archivage et backup
dd

dd

La commande **dd** (device to device) est destinée à la copie physique, bloc par bloc, d'un fichier périphérique vers un fichier périphérique ou quelconque. À l'origine elle était utilisée pour la lecture et l'écriture sur bande magnétique, mais elle peut être employée avec n'importe quel fichier. La commande dd permet de réaliser des copies physiques de disques et de systèmes de fichiers.

Argument	Rôle
if=fichier	Nom du fichier en entrée (celui à copier).
of=fichier	Nom du fichier en sortie.
bs=n	Taille du bloc en octets.
count=n	Nombre de blocs à copier.
skip=n	Nombre de bloc à sauter au début du fichier d'entrée.
conv=	Conversion de l'entrée.
seek=	Nombre de blocs à sauter au début du fichier de sortie.
-s	Shell (commande de connexion) par défaut de l'utilisateur (variable SHELL).
- p	Le mot de passe de l'utilisateur.

Archivage et backup
dd

dd

L'option **conv** admet les paramètres suivants :

- `ascii` : convertir l'EBCDIC en ASCII.
- `ebcdic` : convertir l'ASCII en EBCDIC.
- `block` : compléter les blocs se terminant par un saut de ligne avec des espaces, jusqu'à atteindre la taille mentionnée par bs.
- `unblock` : remplacer les espaces en fin de blocs (de taille cbs) par un saut de ligne.
- `lcase` : transformer les majuscules en minuscules.
- `ucase` : transformer les minuscules en majuscules.
- `noerror` : continuer même après des erreurs de lecture.
- `notrunc` : ne pas limiter la taille du fichier de sortie.
- `sync` : compléter chaque bloc lu avec des NULs pour atteindre la taille ibs.

Ici vous allez placer le secteur de boot de la partition (où est installé lilo ou grub) dans un fichier. Le fichier ainsi créé pourra être utilisé avec le chargeur de NT/2000/XP pour démarrer sous Linux.

```
# dd if=/dev/sda1 of=boot.lnx bs=442 count=1 "
```

Pour créer un fichier vide d'une taille de 1Mo :

```
$ dd if=/dev/zero of=vide bs=1024 count=1024 "
```

Planification des tâches
avec Cron

Présentation

Le service **cron** permet la programmation d'événements à répétition. Il fonctionne à l'aide d'une table, appelée une **crontab**. C'est un fichier texte, éditable avec un simple éditeur, par exemple vi.
Pour modifier votre **crontab** personnelle utilisez la commande crontab pour éditer la table, avec le paramètre `-e`.
Les fichiers crontabs sont sauvés dans `/var/spool/cron`.
Le service **cron** doit tourner pour que les crontabs soient actives.

```
$ ps -ef|grep cron "
```

Planification des tâches
avec Cron

Formalisme

Le format d'un enregistrement de crontab est le suivant :

Minutes	Heures	Jour du mois	Mois	Jour semaine	Commande
1	2	3	4	5	6

Utilisez le format suivant pour les valeurs périodiques :

- Une valeur pour indiquer quand il faut exécuter la commande. Ex : la valeur 15 dans le champ minute signifie la quinzième minute.
- Une liste de valeurs séparées par des virgules. Ex : 1,4,7,10 dans le champ mois pour janvier, avril, juillet, octobre.
- Un intervalle de valeurs. Ex : 1-5 dans le champ jour de la semaine indique du lundi (1) au vendredi (5). Le 0 est le dimanche et le 6 le samedi.
- Le caractère * pour toutes les valeurs possibles. Ex : * dans le champ jour du mois indique tous les jours du ou des mois.

Planification des tâches
avec Cron

Exemple

Exécution de df tous les jours, toute l'année, tous les quarts d'heure :

```
0,15,30,45 * * * * df > /tmp/libre "
```

Exécution d'une commande tous les jours ouvrables à 17 heures :

```
0 17 * * 1-5 fin_travail.sh "
```

Lister les crontabs actives :

```
$ crontab -l"
```

Supprimer la crontab active :

```
$ crontab -r "
```

Éditer la crontab d'un utilisateur particulier :

```
# crontab -u user "
```

Planification des tâches
avec Cron

crontab système

La configuration crontab générale pour le système est dans `/etc/crontab`.

```
SHELL=/bin/bash "
PATH=/sbin:/bin:/usr/sbin:/usr/bin "
MAILTO=root"
HOME=/ "
# run-parts "
```

Ici tous les jours à 4h02 du matin run-parts `/etc/cron.daily` est exécuté. Le script **run-parts** accepte en paramètre un répertoire et exécute tous les programmes présents dans ce répertoire.

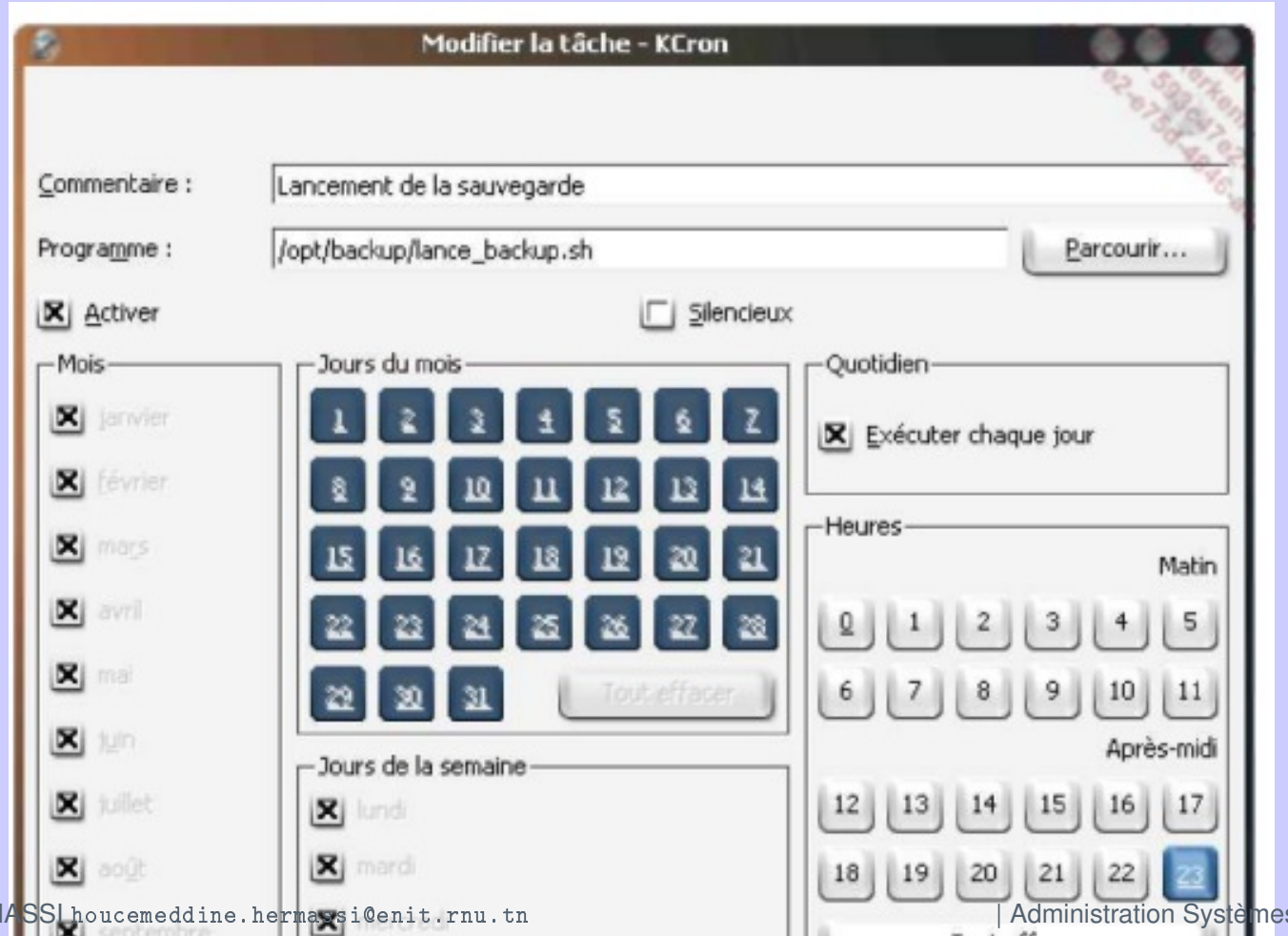
```
$ ls cron.daily/ "
```

Parmi les programmes exécutés, remarquez **logrotate** qui permet d'effectuer des sauvegardes et de renommer des fichiers logs et des journaux du système afin que ceux-ci ne deviennent pas inexploitables à cause de leur taille. Le programme **tmpwatch** est chargé de nettoyer le système des fichiers inutilisés (dans /tmp par exemple).
Enfin, le répertoire /etc/cron.d contient des crontabs supplémentaires.

Planification des tâches
avec Cron

Crontab en mode graphique

Quelques outils permettent d'éditer une crontab de manière visuelle sans passer par un éditeur de texte. L'outil kcron sous KDE est très populaire et très bien adapté pour cela.



Merci pour votre attention!