https://vader-fr.fr/spip.php?article170

Script bash linux d'archivage des journaux avec rotation des archives

- Linux & Logiciels Libres -

Publication date: jeudi 25 février 2016

Ce petit script assez simple recherche, via la commande `find` les fichiers de taille supérieure à celle paramétrée, puis les met dans une archive compressée tar gz.
Si l'archive a bien été crée, le fichier original est vidé.
En cas d'existence d'une ancienne archive, leurs numéros sont incrémentés jusqu'à 9, la neuvième (et donc plus ancienne) archive étant effacée.

Bien sûr, pour les (quelques) logiciels qui ont leur propre système de rotation de journal, cela va archiver et empiler les fichiers journaux et leurs sauvegardes.
Une solution pourrait être de ne pas traiter que les fichiers, mais également les répertoires, en se cantonnant par contre à la racine du répertoire des journaux. (`/var/log`)

De même, un `find` basé sur la date de dernière modification du fichier permettrait de nettoyer les anciennes archives pour les journaux dont le nom évolue avec leur propre système de rotation.

Ce script pourra ensuite être lancé automatiquement par une tâche `cron`, que l'on ajoutera par `crontab -e` en `root`
Auquel cas il ne faudra pas oublier de rediriger les `echo` du script vers un fichier de log, comme `/var/log/logrotate.log`

Exemple d'une définition de tâche cron programmée pour se lancer tous les jours à 20h00

```
# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * command to be executed
 0 20  *  *  * /usr/local/src/logrotate.sh
```

Au début du fichier, on retrouve le *shebang*, indiquant l'interpréteur de commande, ainsi que les principaux paramètres.

```
#!/bin/bash
# script d'archivage des logs de plus de X Mio, avec vidage des logs et rotation des archives
# paramètres du script
rep_log="/var/log";
rep_arch="/media/KastorPollux/Software/Linux/log";
```

```
fichier_liste="/tmp/liste_temp_logrotate"; # gestion des fichiers avec espaces via mise en liste, sinon il faudrait 2 scripts, un de recherche et un de traitement en -exec du find
taille_limite=5; # en Mio, ce qui n'est pas négligeable pour 1) des journaux, donc du texte 2) une multitude de journaux
delai=15; # délai en jours pour effacement des anciennes archives
# fin des paramètres
```

Une petite fonction permettra de gérer les erreurs fatales en affichant un message avant de sortir brutalement du script en revoyant un code d'erreur.

```
# fonction pour les erreurs fatales
erreur(){
 echo "ERREUR : $1">&2;
 exit $2
}
```

Au début du script à proprement parler, un certain nombre de vérifications sont faites :

- le répertoire des fichiers journaux
  - `-d` est il bien un répertoire ? (et donc existe il ?)
  - `-r` est il lisible ?
    - on ne teste que pour la lecture seule car les droits d'effacement du fichier journal ou de son contenu seront vérifiés au cas par cas par la suite.

- le répertoire des archives
  - `-d` est il bien un répertoire ? (et donc existe il ?)
    - si non, il est créé. l'option `-p` de `mkdir` permet de créer les répertoires parents si besoin.
    - En cas d'échec de la création du répertoire, on sort en renvoyant un code d'erreur.
  - `-w` est il modifiable ? il faudra en effet y créer et modifier des fichiers.

- le répertoire où l'on créera la liste temporaire
  - est il modifiable ? (et donc existe il ?)
  - récupéré par la commande `dirname` sur le nom complet du fichier liste indiqué en paramètre, c'est donc forcément un répertoire.

les tests des droits lecture/écriture sont ceux valables pour l'utilisateur en cours.

```
if [ -d "$rep_log" -a -r "$rep_log" ]
then
 # repertoire "$rep_log" existe en tant que repertoire, et on peut le lire
```

```
if [ ! -d "$rep_arch" ]
then
mkdir -p "$rep_arch"
# si le mkdir ne signale pas d'erreur
if [ ! $? -eq 0 ]
then
erreur "Le répertoire $rep_arch n'a pu être créé" 82
fi
fi
if [ ! -w "$rep_arch" ]
then
erreur "Le repertoire $rep_arch n'est pas modifiable" 81
fi
rep_liste=$(dirname "$fichier_liste");
if [ ! -w "$rep_liste" ]
then
erreur "Le repertoire $rep_liste n'est pas modifiable, impossible de créer la liste" 83
fi
```

Si on arrive jusque là, alors tout est en place et les choses devraient bien se passer.
La variable *taille_limite* est entourée d'accolades (`{ ... }`) afin de ne pas embarquer le tout avec le "M" suivant comme nom d'une variable qui n'existe pas.
On envoie le résultat du `find` dans un fichier liste que l'on lit ensuite ligne par ligne.

```
echo "Nettoyage des logs - $(date).";
# comptage de la taille avant
taille_avant=$(du -sh "$rep_log" | awk '{ print $1 }');
# recherche des fichiers de + de X Mio dans $rep_log et archivage par tar / gz (extension tgz) dans
$ep_arch
find "$rep_log" -size +${taille_limite}M 2>/dev/null 1>"$fichier_liste"; # le fichier contient des lignes
de type "$rep_log/nom fichier"
while read ligne;
do
```

Les commandes `basename` et `dirname` permettent de séparer le fichier de son répertoire parent.
Puis on vérifie l'existence d'une archive précédente. [1]
S'il y a déjà une archive, alors il faudra incrémenter le numéro des archives existantes pour ce fichier.
On va donc de 9 à 2, en enlevant la numéro 9 et en incrémentant de 1 le numéro des précédentes.
Ainsi, lors du traitement de la numéro 9, archive.8 devient archive.9, puis à la boucle suivante (8), archive.7 devient archive.8, et ainsi de suite jusqu'à archive.1 qui devient archive.2, libérant le numéro 1.
Le numéro 1 étant libéré, l'archive existante devient archive.1, permettant de créer une nouvelle archive.

```
fichier=$(basename "$ligne"); # extraction du nom du fichier (sans arborescence) dans la ligne
```

```
repertoire=$(dirname "$ligne"); # extraction du répertoire parent
if [ -f "$rep_arch/$fichier.tgz" ]
then
# si une archive existe déjà
for ((suffixe=9;suffixe>=2;suffixe--)){
# on enlève l'archive 9 si elle existe
if [ -f "$rep_arch/$fichier.tgz.$suffixe" -a $suffixe -eq 9 ]
then
rm -f "$rep_arch/$fichier.tgz.$suffixe";
fi
# puis on augmente le numéro des anciennes archives (de 8 à 1) si elles existent
precedent=$(expr $suffixe - 1);
if [ -f "$rep_arch/$fichier.tgz.$precedent" ]
then
mv "$rep_arch/$fichier.tgz.$precedent" "$rep_arch/$fichier.tgz.$suffixe"
fi
}
# puis on renomme l'ancienne archive en 1
mv "$rep_arch/$fichier.tgz" "$rep_arch/$fichier.tgz.1"
fi
```

Une fois l'archive du jour créée avec succès, on tentera de vider ou d'effacer le fichier journal traité.
Le teste repose sur le code retour enregistré (`$?`) après la commande tar.
S'il est égal à 0, cela signifie qu'il n'y a pas eu d'erreur.
Une commande `echo` bien formatée placée entre le `tar` et le test suivant rendrait ledit test complètement inutile, le code retour testé étant le dernier, et donc dans ce cas celui de la commande `echo`.

```
# une fois l'ancienne archive renommée
tar -czf "$rep_arch/$fichier.tgz" "$ligne" 2>/dev/null;
# si le tar ne signale pas d'erreur
if [ $? -eq 0 ]
then
```

Si l'on peut modifier le fichier journal, alors on renvoie un texte vide, venant du périphérique spécial `/dev/null`.
Dans le cas contraire, si le répertoire parent du fichier journal est modifiable, alors on effacera le fichier.
Si ces deux méthodes ne sont pas utilisables, on le signale par un message d'erreur, mais sans pour autant quitter le script.

```
# si le fichier est w
if [ -w "$fichier" ]
then
# vidage du fichier log
cat /dev/null > "$fichier"
```

```
 else
 if [ -w "$repertoire" ]
 then
 # suppr du fichier original
 rm -f "$ligne";
 else
 echo "le fichier $ligne n'a pu être vidé ni effacé" >&2; # affichage d'erreur non fatale
 fi
 fi
 else
 echo "le fichier $ligne n'a pu être archivé" >&2 # affichage d'erreur non fatale
 fi
 done<$fichier_liste
```

En fin de script, on efface le fichier temporaire contenant la liste des journaux à traiter car plus grands que la taille fixée, puis on affiche un bref aperçu de l'espace libéré.

```
 # suppr du fichier temporaire de liste
 rm -f $fichier_liste
 taille_apres=$(du -sh "$rep_log" | awk '{ print $1 }');
 find "$rep_arch" -mtime +$delai -type f -exec rm -f {} \;
 echo -e "Fin de nettoyage des logs\nAvant : $taille_avant, Apres : $taille_apres
\n================================";
else
 erreur "Le repertoire $rep_log n'existe pas ou n'est pas lisible" 80
fi
exit 0
```

Script bash archivage, rotation et nettoyage des logs

---

[1] Deux fichier journaux de même nom (dans des sous répertoires différents de rep_log) donneraient deux archives de même nom, mais ce serait vraiment pas de bol.