

Script

Faire un script Linux pour automatiser ou industrialiser des commandes

- [Faire un script Linux](#)
- [Script de sauvegarde](#)
- [Commande en arrière plan](#)

Faire un script Linux

Comment faire un script Linux ?

Création du fichier

Créer un fichier `monFichier.sh` :

```
nano monFichier.sh
```

L'extension ".sh" n'est pas obligatoire. Elle permet juste d'indiquer le type de fichier.

Contenu du script

Tous les script linux devront commencer par `#!/bin/bash`

Paramètre Shell

Paramètres	Fonction
<code>\$1-\$9</code>	Représente les paramètres positionnels pour les arguments 1 à 9
<code>\${10}-\${n}</code>	Représente les paramètres positionnels pour les arguments après 9
<code>\$0</code>	Représente le nom du script
<code>\$*</code>	Représente tous les arguments comme une seule chaîne de caractères
<code>\$@</code>	Identique au <code>\$*</code> , mais différent lorsqu'il est entouré de (").
<code>\$#</code>	Représente le nombre total d'argument
<code>\$\$</code>	PID du script
<code>\$?</code>	Représente le dernier code retourné

Exécution du script

Pour exécuter un fichier ".sh" :

```
bash monFichier.sh
```

Si vous souhaitez exécuter un script sans utiliser la commande `bash`. Vous pouvez le faire en vous rajoutant les droits d'exécution sur le fichier.

Ce qui donne la commande suivante :

```
chmod +x monFichier.sh
```

Vous pouvez désormais exécuter le script de la manière suivante :

```
./monFichier.sh
```

Exemple script

Voici un exemple d'un script qui affiche des informations :

```
#!/bin/bash

# Ceci est un commentaire en bash

# Afficher le mot "Hello world"
echo "Hello world"

# Afficher le nom du script
echo $0
```

La sortie du script :

```
user@server:/home/user$ ./monFichier.sh
Hello world
./monFichier.sh
```

Script de sauvegarde

Prérequis :

- avoir un accès ssh sur un serveur de sauvegarde
- avoir installé [rsync](#) sur la machine et le serveur
- avoir installé zenity

Fonctionnalités :

- juste besoin de lister les dossiers et fichiers à sauvegarder dans un fichier texte une fois le script configuré
- vérifie l'état de la connexion et fait un ping sur le serveur avant la sauvegarde
- crée un log hebdomadaire des fichiers sauvegardés
- affiche une jolie notification à la fin de la sauvegarde
- si on clique sur l'icone de notification, affiche la liste des fichiers sauvegardés

Voici le contenu du script

```
#!/usr/bin/env bash
# rsync from this host to the backup server

#options de sauvegarde manuelle interactive
if [ "$*" = "-m" -o "$*" = "--manual" ]; then
    MODE="Manual"
else
    MODE="Automatic"
fi

#vérification de la connection réseau et boucle en attente de connexion
# eth est à adapter en fonction du nom de votre carte réseau, ici eth pour eth0 + eth1
(ethernet ou wifi)
ETH="eth"
WAIT=0
while [ `ifconfig | grep -A1 ${ETH} | sed -n 's/.*adr:\([0-9]*\.[0-9]*\.[0-9]*\.[0-9]*\).*\/\1/p' | wc -c` = 0 ]; do
    WAIT=1
    sleep 5
```

```

echo waiting
done
#laisse un peu de temps si la connexion était coupée
if [ ${WAIT} = 1 ]; then
sleep 5
fi

#définition des variables
#-----
SCRIPT_PATH=~/scripts/backup##chemin du dossier contenant le script
SRC_FOLDER=~/#####chemin du dossier local contenant les fichiers à sauvegarder
SAVELIST=${SCRIPT_PATH}/savelist#fichier texte contenant la liste des fichiers et sous-
dossiers de SRC_FOLDER à sauvegarder
#-----
USER=nom_utilisateur####nom d'utilisateur ssh sur le serveur
REMOTE_HOST=server.domain.fr##adresse distante du serveur de sauvegarde
ALT_HOST=server.local####adresse locale du serveur de sauvegarde
DST_FOLDER=backup####nom du dossier de sauvegarde sur le serveur
#-----
NICE=5####priorité donnée au script
#-----

#teste si on est en local, auquel cas changement de nom de serveur
ping -c 2 -d ${ALT_HOST} && REMOTE_HOST=${ALT_HOST}

LOGIN=${USER}@${REMOTE_HOST}
DATE=`date +%k:%M`

#création du dossier de log si inexistant
if [ ! -e ${SCRIPT_PATH} ] ; then mkdir ${SCRIPT_PATH}/log; touch ${SCRIPT_PATH}/log/last ;
fi

#fonction de notification
notification() {
## ${1} = "terminée" ou "annulée"
## ${2} = durée d'affichage en secondes
echo message:"${DATE}\nLa sauvegarde sur ${REMOTE_HOST} est ${1}" | zenity --notification --
listen --window-icon="info" | zenity --notification --timeout=${2} --window-icon="info" --
text="Sauvegarde sur ${REMOTE_HOST} ${1} (${DATE})" && if cat ${SCRIPT_PATH}/log/list | grep
""; then cat ${SCRIPT_PATH}/log/list | grep -v /$ | zenity --text-info --title="Fichiers

```

```

synchronisés" --width=800 --height=600; else zenity --info --text="Aucun fichier synchronisé";
fi

}

#fonction de sauvegarde et log
save() {
  #synchronisation et log
  {
    echo "# ${MODE} backup"
    echo "# Starting: "`/bin/date`
    nice -n ${NICE} rsync -avrz --delete \
    --files-from=${SAVELIST} \
    ${SRC_FOLDER} \
    ${LOGIN}:${DST_FOLDER}
    echo "# Ending: "`/bin/date`
    echo "# -----"
    echo
  } >> ${SCRIPT_PATH}/log/`date +%G-%V`.log
  date >> ${SCRIPT_PATH}/log/last
  #sauvegarde du nom des derniers fichiers synchronisés
  tac ${SCRIPT_PATH}/log/`date +%G-%V`.log | sed -n '2,/#\ -/p' | tac | sed -e '1,5d' | sed -e
:a -e '$d;N;2,5ba' -e 'P;D' >${SCRIPT_PATH}/log/list
  #notification
  notification "terminée" 1200
}

#dialogue si mode interactif
if [ ${MODE} = "Automatic" ]; then
  save
else
  zenity --question --title="Sauvegarde Manuelle sur ${REMOTE_HOST}" --text="Sauvegarder sur
${REMOTE_HOST} maintenant?

  La dernière sauvegarde date du :
  `tail -1 ${SCRIPT_PATH}/log/last | sed 's/ (UTC.*//`
  Les modifications ultérieures seront transférées sur ${REMOTE_HOST}.
  "

  if [ $? = "0" ]

```

```

then
    save
else
    notification "annulée" 60
fi
fi

```

rendez ensuite le script exécutable

```

chmod u+x /chemin/vers/backup.sh

```

adaptez la partie du script contenant les variables à votre configuration

```

#définition des variables
#-----
SCRIPT_PATH=~/.scripts/backup##chemin du dossier contenant le script
SRC_FOLDER=~/.##chemin du dossier local contenant les fichiers à sauvegarder
SAVELIST=${SCRIPT_PATH}/savelist##fichier texte contenant la liste des fichiers et sous-
dossiers de SRC_FOLDER à sauvegarder
#-----
USER=nom_utilisateur##nom d'utilisateur ssh sur le serveur
REMOTE_HOST=server.domain.fr##adresse distante du serveur de sauvegarde
ALT_HOST=server.local##adresse locale du serveur de sauvegarde
DST_FOLDER=backup##nom du dossier de sauvegarde sur le serveur
#-----
NICE=5##priorité donnée au script
#-----

```

et créez le fichier contenant la liste des dossiers à sauvegarder (ici /chemin/vers/savelist)
 /\ les fichiers et dossiers listés sont des sous-fichiers/dossiers du dossier défini auparavant par SRC_FOLDER (par défaut le home)
 /\ attention à la casse des caractères

```

Documents/Travail
Images/Travail
autres/dossier_super_important

```

Pour lancer le script

```

/chemin/vers/backup.sh

```

pour le lancer avec une boîte de dialogue de confirmation (j'ai fait un lanceur sur le tableau de bord avec la commande "bash /chemin/vers/backup.sh --manual")

```
/chemin/vers/backup.sh --manual
```

et enfin pour automatiser les sauvegardes, il faut utiliser [cron](#)

```
crontab -e
```

et ajouter les paramètres suivants pour lancer le script à midi du lundi au vendredi (le >>/chemin/vers/log/cronlog 2>&1 permet de créer un log pour debugger)

```
DISPLAY=":0.0"
USER="nom_utilisateur"
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
# m h dom mon dow    command
00 12 * * 1-5 /chemin/vers/backup.sh >>/chemin/vers/log/cronlog 2>&1
```

Commande en arrière plan

Comment exécuter une commande en tâche de fond ?

Exécuter une commande Linux en arrière-plan

Pour exécuter une commande en arrière-plan, ajoutez le symbole esperluette (`&`) à la fin de la commande :

```
command &
```

L'ID du travail shell (entouré de crochets) et l'ID du processus seront imprimés sur le terminal :

```
# Output  
[1] 25177
```

Cependant le processus en arrière-plan continuera à écrire des messages sur le terminal à partir duquel vous avez appelé la commande.

Pour supprimer les messages `stdout` et `stderr`, utilisez la syntaxe suivante :

```
command > /dev/null 2>&1 &
```

Screen ?