


Linux - Cheatsheet

🕒 Created	@December 18, 2022 2:00 PM
🏷️ Tags	Linux Systeme
👤 Created by	 Rudy Hilarion-oris
🌟 Status 1	In progress

Quelques commandes Unix

```
$ apropos      # Liste les pages du manuel concernant un sujet
$ date        # affiche la date du jour
$ cal         # affiche le calendrier
$ whoami      # affiche le nom de l'utilisateur
$ passwd      # permet de changer son propre mot de passe
$ uname       # affiche le nom du système
$ uname -a    # affiche toutes (option -a) les informations du système
$ echo texte  # affiche le texte « texte » passé en argument de la commande
$ cat fichier # affiche le contenu du fichier «fichier» passé en argument
$ more fichier # affiche le contenu du fichier «fichier» passé en argument
$ less fichier # affiche le contenu du fichier «fichier» passé en argument et permet sa « pagination »
$ umask -S    # affiche le masque utilisateur sous forme symbolique
$ umask       # affichage du masque en mode numérique
$ umask droits # permet de redéfinir (temporairement) le masque (ex : umask u=,g=,o=, umask 777)
$ exit        # quitte l'interface du terminal
$ man         # affiche le manuel pour des commandes
$ who         # affiche qui est connecté
$ mount       # affiche les systèmes de fichiers qui sont montés
$ uptime      # affiche la disponibilité
$ wc -l /etc/passwd # compte le nombre de lignes dans le fichier "passwd"
$ wc -lw /etc/passwd # compte le nombre de lignes et de caractères dans le fichier "passwd"
$ sleep 60    # Attendre 60 secondes
$ ntpdate time.nist.gov # Mettre à l'heure ntp
$ time trouve_prince_charmant beau riche # Mesurer le temps pris par une commande
```

Les bases

Le système des fichiers et les droits standards

```
# Syntaxe
$ ls [ OPTIONS ] # Liste les fichiers «ordinaires» (ne commençant pas par .) dans le rép. courant

# Options :
# -l : Afficher une liste détaillée (long)
# -a : Liste tous (all) les fichiers dans le rép. courant
# -t : Liste tous les fichiers en les triant par date (time)
```

```
# -S : Liste tous les fichiers en les triant par taille (size)
# -r : Liste tous les fichiers en inversant (reverse) l'ordre de tri
```

```
$ mkdir rép                # Créer un répertoire
$ mkdir -p rép1/rép2       # Créer des répertoires imbriqués

$ cd nouveau_rép           # Change de répertoire
$ cd ..                    # Répertoire parent
$ cd                        # Répertoire personnel
$ cd ~alice                 # Répertoire personnel de alice

$ cp fichier_orig fichier_dest # Copier un fichier vers un autre
$ cp fichier1 fichier2 rép     # Copier des fichiers dans un répertoire
$ cp [-r] source destination  # copie de fichiers
$ cp -r rép_orig rép_dest     # Copier des répertoires entiers (recursively)

$ rsync a rép_orig/ rép_dest/  # Se comporte comme la précédente commande, mais plus rapide
$ mv fichier_orig fichier_dest # Renommer un fichier, lien ou répertoire

$ rm [-r] fichier           # Effacer fichier (ou répertoire avec option -r )
$ rm fichier1 fichier2      # Supprimer (remove) des fichiers ou des liens
$ rmdir rép                 # Supprimer un répertoire (remove dir)
$ rm -rf rép                # Supprimer un répertoire non vide (force)

$ stat fichier              # Afficher les méta-données du fichier fichier
$ touch fic                 # « création » d'un fichier fic - modification horodatage
$ diff fic_a.txt fic_b.txt  # Affiche les différences entre les deux fichiers
$ vi fic.txt                # Edition du fichier fic.txt
$ pwd                      # Afficher répertoire courant (print working dir)
```

Lien symbolique et physique

```
# Syntaxe
$ ln -s cible lien # Lien symbolique
                    # Ressemble aux « raccourcis »
                    # Fichier qui « pointe » vers un autre fichier (cible)
                    # Si le fichier cible est supprimé le lien symbolique est cassé !

# Exemple
$ cp /etc/hosts ~/hosts1 # copie les elements de /etc/hosts dans un fichier du répertoire courant
$ ln -s hosts1 hosts1.lnk # creer un lien symbolique à vers le fichier hosts1.lnk
# Le lien symbolique est représenté par une flèche "->" vers le fichier qu'il pointe
```

```
# Syntaxe
$ ln cible lien # Lien physique
                # Le nom pointe directement vers le contenu mémoire
                # Le contenu du fichier est supprimé qu'après suppression de tous les liens physiques qui s'y rapportent
```

La sécurité des fichiers : les droits



chmod : cette commande permet de changer les permissions des fichiers sur linux.

```
# Syntaxe 1
# Options : les paramètres de la commande par exemple +R pour rendre récursif
# MODE : les permissions à expliquer soit en octal (texte), soit en décimale (chiffre)
# fic : le nom du fichier
$ chmod [OPTIONS] MODE fic

# Syntaxe 2 : changer des permissions à l'aide de la notation de texte
$ chmod [OPTIONS] [ u g o a ] [ - + = ] [ r, w, x ] fic

# Options
# -c : signale uniquement lorsqu'une modification est apportée
```

```
# -f : supprimer la plupart des messages d'erreur
# -R : modifier les fichiers et les répertoires de manière récursive
```

```
# Exemples :

# ---- [ Ajouter des permissions ]
# Attribut droits lecture / écriture au propriétaire (user, read, write)
# Ajouter droits d'exécution aux à tous (all)
$ chmod u=rw,a+x fic
# Attribut tous les droits lecture / écriture / exécution à tous (user, group, other)
$ chmod u=rwx,g=rwx,o=rwx fic
$ chmod 777 fic # Attribut tous les droits sur fic à User, Group et Other
$ chmod u+w fic # Ajouter droits en écriture au propriétaire (user, write)
$ chmod g+r fic # Ajouter droits en lecture au groupe du fichier (group, read)
$ chmod o+x fic # Ajouter droits d'exécution aux autres utilisateurs
$ chmod a+rw fic # Ajouter droits lecture / écriture à tous (all)
$ chmod a+rx * # Rendre fic. exécutables exécutables par tous

# ---- [ Supprimer les permissions ]
# Retire les permissions de lecture et d'exécution pour le propriétaire du fichier
$ chmod u-rx fic
# Retire les permissions de modification et d'exécution pour le groupe du fichier
$ chmod g-wx fic
# Retire toutes les permissions pour les autres utilisateurs
$ chmod o= fic

# ---- [ Autres ]
# Rendre le répertoire et tous les fichiers qu'il contient accessibles
# par tous les utilisateurs (recursive)
$ chmod -R a+rx rép
# Changer le propriétaire et le groupe d'un répertoire et tout ce qu'il contient
$ chown -R nouvproprio:nouv groupe rép
```

Numéros d'autorisation de fichier

Permissions	Valeurs	Descriptions
<code>rwX</code>	7	Accorder tous les droits lecture / écriture / exécution
<code>rw-</code>	6	Accorder uniquement les droits de lecture / écriture
<code>r-X</code>	5	Lecture, exécution
<code>r--</code>	4	Lecture (read)
<code>-W-</code>	2	Ecriture (write)
<code>--X</code>	1	Exécution (execute)
<code>---</code>	0	Aucun accès

Classes	Opérations	Permissions
user (u)	Assignation (u)	Lecture (r)
group (g)	Retirer (-)	Ecriture (w)
other (o)	Ajouter (+)	Exécution (x)
all (a)		



Pour les débutant, il est conseiller d'écrire les permissions sous ce format :

```
chmod u=rwx,g=rwx,o=rwx fic
```

```
# Changement du propriétaire du ou des fichiers ou répertoires listés dans la commande "chown".
$ chown nom_utilisateur [fic | rep]
```

```
# Positionne des droits sur les actions de l'utilisateur durant sa session.
$ umask [droits]

# Exemples :
$ umask 000 # (ne fait rien)
$ umask 022 # (rw-r--r--)
$ umask 027 # (rw-r-----)

# Retire les permissions de lecture et d'exécution pour le propriétaire du fichier
$ umask u-rx
# Retire les permissions de modification et d'exécution pour le groupe du fichier
$ umask g-wx
# Retire les permissions de modification et de lecture pour les autres utilisateurs
$ umask o-rw
```



Notes :

Il est important de noter que la commande "umask" ne modifie pas les permissions des fichiers et répertoires existants, mais uniquement celles des nouveaux fichiers et répertoires créés par la suite. Par défaut, les droits pour utilisateur sont à 777 (rwxrwxrwx) pour les répertoire, 666 (-wx-wx-wx) pour les fichiers.

Lien utile

Chmod 656

View (u)ser, (g)roup and (o)thers permissions for chmod 656 (chmod a+rwx,u-x,g-w,o-x) or use free online chmod calculator to modify permissions easily.

<https://chmodcommand.com/chmod-656/>

The screenshot shows the Chmod Command Calculator interface. It has three columns for permissions: Owner Rights (u), Group Rights (g), and Others Rights (o). Each column has checkboxes for Read (r), Write (w), and Execute (x). Below these columns, there are radio buttons for 'Recursive' and 'Sticky Bit'. The 'Recursive' option is selected. At the bottom, there is a text input field for the 'Chmod Command' and a 'Calculate' button.

La substitution des commandes



Le Bash propose plusieurs mécanismes de substitution qui permettent de manipuler et de transformer des chaînes de caractères. Voici quelques exemples de substitution couramment utilisés :

1. La substitution de commande

La substitution de commande permet d'exécuter une commande et de remplacer son résultat par sa sortie

```
$ echo "Il y a $(ls | wc -l) fichiers dans ce répertoire."
```

2. La substitution de variable

La substitution de variable permet de remplacer le nom d'une variable par sa valeur.

```
$ echo "Ma maison se trouve à ${MAISON}."
```

3. La substitution de commande avec l'opérateur ":"

L'opérateur ":" permet de définir une valeur par défaut pour une variable si celle-ci n'est pas définie.

```
$ echo "Ma maison se trouve à ${MAISON:-inconnue}."
```

4. La substitution de sous-chaîne

La substitution de sous-chaîne permet de remplacer une partie d'une chaîne de caractères par une autre chaîne.

```
$ echo "${Bonjour:0:1}onjour"
```

Édition de fichiers

Éditeurs	Exemples
Graphique	gedit, kate / kwrite, geany
Historiques	nano, emacs
Non graphique et ligne de commande	vi, vim

```
# Exemples
$ gedit /home/stag/fichier.txt
$ nano ~/fichier.txt
$ emacs ~/fichier.txt
$ vi ~/fichier.txt
$ vim ~/fichier.txt
```

Les Processus

Les information sur les processus



ps - permet de retrouver toutes les informations utiles sur un processus.

```
# Syntaxe :
$ ps [ options ]

# Options :
# -e : indique le statut de tous les processus
# -f : tous les informations possibles sur les processus actifs
# -t : affiche les processus actifs liés à un terminal
# -u : affiche les processus actifs liés à un utilisateur.
```

```
# Exemples :
# Recherche parmi l'ensemble des processus, le processus qui port le nom 'firefox'
$ ps -aux | grep -i 'firefox'
# Afficher les informations sur les processus en cours d'exécution pour l'utilisateur courant
$ ps -u $USER
# Afficher les infos sur tous les processus, y compris ceux qui sont cachés
$ ps -e
# Afficher les infos sur tous les processus en cours d'exécution sous forme de liste détaillée
$ ps -ef
# Afficher les infos sur tous les processus en cours d'exécution dans un format spécifique
$ ps -eo pid,user,%cpu,%mem,command
```

Noms	Descriptions
UID	Uid (n° d'identifiant) de l'utilisateur
PID	N° d'identification du processus
PIDP	N° d'identification du processus père
C	Utilisation du processeur pour le sheduler
STIME	Instant de démarrage du processus (H-min-sec). Si le processus a plus de 24 h, il est donné en mois et jours
TTY	Terminal de contrôle du processus
TIME	Temps CPU utilisé en secondes
COMD	Ligne de commande

```
# Obtenir des informations sur les processus
$ pstree [-p] [-l] [-s]
$ htop      # Classement en direct des processus
$ top       # Classement en direct des processus
$ free      # Afficher la mémoire libre
$ vmstat n # Afficher la mémoire virtuelle toutes les n secs:
```

Interaction avec des processus

```
# Syntaxes
$ kill [-s signal_name] pid

$ kill -l [exit_status]
$ kill -signal_name pid ...
$ kill -signal_number pid ...

$ kill [ -SIGTERM | -15 ] PID # Demande l'arret du processus
$ kill [ -SIGKILL | -9 ] PID  # Demande au processus de se terminer
$ kill -9 -1                  # Tue tous processus que l'on a le droit de tuer
$ killall name                # Tue tous les processus qui commence par le nom
$ xkill                       # Tue une application en mode graphique
$ pkill nom_processus         # Tue un processus par son nom

# Exemples :
$ kill %numero_job # Terminer un job en arrière-plan en indiquant son n° de job
$ kill %+          # Terminer le job courant
$ kill %-          # Terminer le job précédent
```

Jobs, fg et bg

```
$ gedit & # lance en arrière plan l'application gedit

$ jobs # affiche les processus lancés par Bash (liste avec des « jobs ID »)
$ fg   # sert à placer un processus en premier plan,
$ bg   # sert à placer en arrière plan un processus endormi (et donc le réveille).
       # fg et bg utilisent les « jobs ID »
```

nice et renice

```
# Gestion des priorités des processus (de -20 à +19)
# -20 : très forte à 19 : très faible
# 0 : priorité par défaut (ou celle du père)
$ nice -n 19 commande # nouveau processus
$ renice -n 19 PID    # processus déjà lancé
```

Autres

```
# Récupération des identifiants d'un processus
$ echo $BASHPID
$ echo $$
$ echo $PPID
```

Quelques raccourcis

Numéro	Noms	Descriptions	Forme 1	Forme 2
1	SIGHUP	Instruction (HANG UP) - Fin de session		
2	SIGINT	Interruption	Ctrl + C	^C
3	SIGQUIT	Instruction (QUIT)	Ctrl + \	^\
4	SIGILL	Instruction illégale		
5	SIGTRAP	Trace trap		
6	SIGABRT (ANSI)	Instruction (ABORT)		
6	SIGIOT (BSD)	IOT Trap		
7	SIGBUS	Bus error		
8	SIGFPE	Floating-point exception - Exception arithmétique		
9	SIGKILL	Instruction (KILL) - termine le processus immédiatement	Ctrl + U	^U
10	SIGUSR1	Signal utilisateur 1		
11	SIGSEGV	Violation de mémoire		
12	SIGUSR2	Signal utilisateur 2		
13	SIGPIPE	Broken PIPE - Erreur PIPE sans lecteur		
14	SIGALRM	Alarme horloge		
15	SIGTERM	Signal de terminaison		
16	SIGSTKFLT	Stack Fault		
17	SIGCHLD ou SIGCLD	modification du statut d'un processus fils		
18	SIGCONT	Demande de reprise du processus		
19	SIGSTOP	Demande de suspension imblocuable	Ctrl + Z	^Z
20	SIGTSTP	Demande de suspension depuis le clavier		
21	SIGTTIN	lecture terminal en arrière-plan		
22	SIGTTOU	écriture terminal en arrière-plan		
23	SIGURG	évènement urgent sur socket		
24	SIGXCPU	temps maximum CPU écoulé		
25	SIGXFSZ	taille maximale de fichier atteinte		
26	SIGVTALRM	alarme horloge virtuelle		

Numéro	Noms	Descriptions	Forme 1	Forme 2
27	SIGPROF	Profiling alarm clock		
28	SIGWINCH	changement de taille de fenêtre		
29	SIGPOLL (System V)	occurrence d'un évènement attendu		
29	SIGIO (BSD)	I/O possible actuellement		
30	SIGPWR	Power failure restart		
31	SIGSYS	Erreur d'appel système		
31	SIGUNUSED			

Flux et Filtres

Les flux

Descriptions	Flux	Alternative	Exemples
Flux d'entrée	0	<code>STDIN</code>	<code>></code> ou <code>>></code>
Flux de sortie standard	1	<code>STDOUT</code>	<code>1></code> ou <code>1>></code>
Flux de sortie d'erreur	2	<code>STDERR</code>	<code>2></code> ou <code>2>></code>

```
# Redirection de l'entrée depuis un fichier
$ cat < /etc/passwd

# Redirection d'une sortie vers un fichier
$ cat /etc/passwd 1> fic # redirige (copie) les éléments de "passwd" dans "fic"
$ echo toto 1>> fic      # redirige le flux d'entree les éléments de "toto" dans "fic"
$ ls rep 2> erreur.log   # redirige le flux de sortie d'erreur de "rep" dans "erreur.log"

# Entrelacement des 2 sorties
$ ls rep 1> fic_resultats 2>&1
$ ls rep &> fic_resultats
```

Les filtres

```
# Afficher l'ensemble du flux passé en entrée :
$ cat fic
$ more fic
$ less fic

# Afficher les n premières / dernières lignes du flux en entrée
$ head [ -n ] fic
$ tail [ -n ] fic

# Sélectionner les lignes du flux contenant un « motif » donné
$ grep "root" /etc/passwd

# Extraire des champs (le 1er et le 3ème ici) d'un flux:
$ cut -d ":" -f 1,3 fic

# Compter le nombre de lignes / mots / caractères
$ wc /etc/services
```




uniq : tri sur un fichier ne ramenant qu'une occurrence unique d'une ligne donnée dans le fichier à trier.

Syntaxe

\$ **uniq** [-udc] fic

Options :

- # -u : ramène, en résultat du tri, des lignes unique,
- # -d : liste que les lignes dupliquées,
- # -c : liste toutes les lignes + nombre d'occurrence(s) de la ligne affichée.



sort : tri des lignes d'un fichier sur des critères de tri donné

Syntaxe :
sort [-r] [-o fic_out] [-t car] [-k num_champ[n]] fic.txt

Options :
-n : tri sur champd numérique (?),
-u : ramène des lignes uniques
-t : caractère séparateur des champs
-k : numero du champ

Exemples :
\$ sort fic.txt # Trie un fichier dans l'ordre croissant

\$ sort -u fic.txt # Affiche qu'une fois les lignes identiques
\$ sort -r fic.txt # Trie un fichier dans l'ordre inverse
\$ sort -b fic.txt # Trie un fichier en ignorant les espaces blancs de début et de fin
\$ sort -f fic.txt # Trie un fichier en ignorant la casse

Trie un fichier en fonction du deuxième champ (délimité par un espace)
\$ sort -k 2 file.txt
Trie un fichier en fonction du deuxième champ dans l'ordre inverse
\$ sort -k 2,2 -r fic.txt
Trie numérique sur un fichier en fonction du 4eme champ, délimité par ":"
\$ sort -n -t":" -k4,4 fic.txt
Trie un fichier et supprimer les lignes en double
\$ sort fic.txt | uniq
Trie un fichier et fusionner les lignes qui ne diffèrent que par la casse
\$ sort -f fic.txt | uniq

La recherche de fichiers

Find



find - permet de rechercher des fichiers et des répertoires en fonction de divers critères.

Syntaxe :
\$ find chemin [option(s)] [action(s)]

Options :
-name : reconnaît les métacaractères du bash, -o -a

```
# -type          : f (fichier), d (repertoire) ou l (lien)
# -user -group
# -mtime -atime -ctime : -mtime -12 (en jours)
# -size          : -size +30k
# -maxdepth / -mindepth

# Actions :
# -print (action par défaut)
# -delete
# -exec ex : -exec commande {} \;
```

```
# Exemples :
# Recherche de fichier par nom
# Trouve tous les fichiers qui ont pour nom 'file.txt'
$ find . -name 'file.txt'
# Trouve tous les fichiers en .txt dans le répertoire
$ find . -name '*.txt'

# Recherche de fichiers par type
# Trouve tous les répertoires
$ find . -type d
# Trouve tous les liens symboliques
$ find . -type l

# Recherche de fichiers par taille
# Trouve tous les fichiers ayant un poid supérieurs à 100 MB
$ find . -size +100M
# Trouve tous les fichiers ayant un poid inférieur à 1 KB
$ find . -size -1k

# Recherche de fichiers par heure de modification
# Trouver tous les fichiers modifiés au cours de la dernière journée
$ find . -mtime -1
# Trouver tous les fichiers modifiés il y a plus de 30 jours
$ find . -mtime +30

# Combinaison de plusieurs critères
# Trouve tous les fichiers '*.txt' modifiés au cours de la semaine dernière
# et dont la taille est supérieure à 50 Ko
$ find . -name '*.txt' -mtime -7 -size +50k
# Trouve tous les fichiers en .sh dans rép et exécute une commande sur chacun
$ find . -name "*.sh" -exec cat {} \;
# Recherche tous les fichiers dans le répertoire courant (.) avec log dans leur nom
$ find . -name "*log*"
# Trouve tous les fichiers en .pdf dans rép et exécute une commande sur chacun
$ find . -name "*.pdf" -exec xpdf {} ';' ;'
```

locate



locate - permet de rechercher rapidement des fichiers et des répertoires sur votre système par leur nom.

```
# Syntaxe :
$ locate [ Option(s) ] Fichier

# Options :
# -e Afficher que les fichiers existants dans le système
# -c Count : compter le nombre d'entrées)
# -regex ou -r Faire une recherche en utilisant les regex
# -i Ignorer la casse (rendre locate insensible à la casse)
# -l <Num> Réduire le nombre d'entrée du résultat à Num
```

```
# Exemples :
$ locate "*bar*" # Recherche rapide dans tout système
$ locate ".conf" # Recherche tous les fichiers avec l'extension .conf
$ locate "*.extf" # Recherche tous les fichiers avec une extension particulière
$ locate fichier* # Trouve les fichiers qui commencent par le mot fichier

# Trouve tous les répertoires qui portent le nom "documents"
```

```
$ locate -d documents
# Trouve tous les répertoires et fichiers qui sont dans le répertoire "/home"
$ locate -b "/home"
# Trouve tous les fichiers et répertoires qui comporte le mot "exemple" dans leur nom
$ locate -r "exemple"
```

Note : Les fichiers qui ont été récemment créer ne serait pas encore enregistrer dans la base de données de Locate. La base de données de locate est mise à jour au min une fois par jour.

Il est possible de mettre à jour cette base de données en utilisant la commade : `updatedb`

grep



grep - permet de rechercher un / des patterne(s) dans un fichier

```
# Syntaxe
$ grep [OPTIONS] MOTIF [FICHIER...]
$ grep [OPTIONS] [-e MOTIF | -f FICHIER] [FICHIER...]

# Options
# -i : Recherche un terme en ignorant la casse
# -r : Recherche récursive
# -v : Inverse la recherche
# -o : Affiche seulement les parties (non vide) correspondantes d'une ligne
# -n : Affiche les lignes sélectionnées et leur n° de ligne

# Exemples :
# Recherche un motif dans un fichier
$ grep pattern fic
# Recherche un motif dans plusieurs fichiers
$ grep pattern fic1.txt fic2.txt fic3.txt
# Recherche un motif dans tous les fichiers d'un répertoire
$ grep pattern *

# Recherche un motif dans tous les fichiers d'un répertoire et de ses sous-répertoires
$ grep -r pattern .
# Recherche un motif et affiche les numéros de ligne des lignes correspondantes
$ grep -n pattern fic.txt
# Recherche un motif et affiche le contexte des lignes correspondantes
$ grep -C 3 pattern fic.txt
# Recherche un modèle et affiche uniquement les noms des fichiers qui contiennent une correspondance
$ grep -l pattern fic1.txt fic2.txt fic3.txt
# Recherche un modèle et affiche uniquement les lignes qui ne contiennent PAS de correspondance :
$ grep -v pattern fic.txt
```

whereis



whereis - utilisée pour localiser les fichiers binaires, source et de page de manuel pour une commande donnée.

```
# Syntaxe
$ whereis [options] commande

# Options
# -l : Rechercher les répertoires dans lesquels la commande whereis recherche.
# -p : Rechercher uniquement les binaires de commande
# -s : Rechercher uniquement les fichiers source
# -m : Permet de rechercher uniquement les fichiers man
```

```
# Exemples :
# Localise les fichiers binaires, source et de page de manuel pour la commande "ls"
$ whereis ls
# Recherche uniquement le fichier binaire pour la commande "ls"
$ whereis -b ls
# Recherche uniquement le fichier de la page de manuel de la commande "ls"
$ whereis -m ls
# Localise les fichiers binaires, source et de page de manuel pour toutes les commandes dans le /usr/bin répertoire
$ whereis -b -s -m *
```

Archivage et Compression

```
# Syntaxes
$ tar -c[v][P][f archive] fichier # Création d'archive
$ tar -t[v][f archive]             # Liste des fichiers archivés
$ tar -x[v][P][f archive]          # Extraction de l'archive

# Options de tar :
#   c : créer
#   t : tester / lister
#   x : extraire
#   j : (dé)compression bzip2 / bunzip2 à la volée
#   z : (dé)compression gzip / gunzip à la volée
#   J : xz / unxz
```

```
# Exemples :
# Créer une archive compressée
tar -jcvf archive.tar.bz2 /home/stag
tar -zcvf archive.tar.gz /home/stag

# Tester (lister) une archive compressée
tar -jtvf archive.tar.bz2
tar -ztvf archive.tar.gz

# Extraire les fichiers d'une archive compressée
tar -jxvf archive.tar.bz2
tar -zxvf archive.tar.gz
```

```
# Exemples : manipuler des archives zip
$ zip -r archive.zip <files> # créer
$ unzip -t archive.zip       # tester / lister
$ unzip archive.zip          # extraire
```

Planification de tâches



Il existe trois méthodes pour ajouter un cron :

1. En modifiant un des dossiers `/etc/cron.*`
2. avec `crontab -e` qui édite le fichier cron de l'utilisateur stocké dans `/var/spool/cron/crontabs/`
3. Le fichier `/etc/crontab` stocke la configuration de cron.

```
# Syntaxe Crontab
* * * * * command(s)
- - - - -
| | | | |
| | | | | ----- Jour de la semaine (0 - 7) (Dimanche=0 or 7)
```

```

| | | ----- Mois (1 - 12)
| | ----- Jour du mois (1 - 31)
| ----- Heure (0 - 23)
----- Minute (0 - 59)

# Les opérateurs :
# * : signifie n'importe quelle valeur ou toujours
# , : permet de spécifier une liste de valeurs à répéter (ex : 1,3,5 => 1h, 3h, 5h)
# - : permet de spécifier une plage de valeurs
# / : permet de spécifier des valeurs qui seront répétées sur un certain intervalle entre elles

# Les macros prédéfinies
# @yearly - Exécute la tâche spécifiée une fois par an à minuit le 1er janvier. ( = 0 0 1 1 * )
# @monthly - Exécute la tâche spécifiée une fois par mois à minuit le premier jour du mois. ( = 0 0 1 * * )
# @weekly - Exécute la tâche spécifiée une fois par semaine à minuit le dimanche. ( = 0 0 * * 0 )
# @daily - Exécute la tâche spécifiée une fois par jour à minuit. ( = 0 0 * * * )
# @hourly - Exécute la tâche spécifiée une fois par heure au début de l'heure. ( = 0 * * * * )
# @reboot - Exécute la tâche spécifiée au démarrage du système.

```

```

# Pour soumettre une action à exécuter il faut que l'utilisateur en ait la permission
# fichiers **/etc/cron.allow** ou /et **/etc/cron.deny**

$ crontab -e # édition
$ crontab -l # visualisation (list)
$ crontab -r # suppression de toutes les tâches (remove)

```

```

# Quelques exemples du fichier Crontab
..
# m      h      dom      mon      dow      command
# 0      0      1      *      *      ls /etc > ~/fichier.txt

>> à 12:00 AM , le 1er jour du mois on lance la commande ls /etc > ~/fichier.txt

..
# m      h      dom      mon      dow      command
# */5    *      *      *      *      cat /etc/passwd > passwd.txt

>> tous les 5 minutes, on lance la commande cat /etc/passwd > passwd.txt

```

Liens utiles

Cron expression generator by Cronhub

Cron expression generator by Cronhub

 <https://crontab.cronhub.io/>

Les variables d'environnements



Le Bash utilise des variables d'environnement pour stocker des informations sur le système et sur les préférences de l'utilisateur. Vous pouvez afficher la liste des variables d'environnement en utilisant la commande `printenv`.

Commandes	Descriptions
<code>env</code>	Affiche les variables environnements
<code>echo \$NAME</code>	Appel une variable d'environnement par son nom
<code>export NAME=value</code>	Créer une variable
<code>\$PATH</code>	Affiche le chemin de recherche

Commandes	Descriptions
<code>\$HOME</code>	Affiche le répertoire d'accueil
<code>\$SHELL</code>	Le shell courant
<code>export LC_ALL=</code>	Efface le contenu de la variable
<code>unset LCALL</code>	Supprimer complètement une variable d'environnement
<code>\$MANPATH</code>	Liste de dossiers où le système doit chercher les pages de manuel.

Quelques variables de paramètres régionaux

Commandes	Descriptions
<code>LANG</code>	Le paramètre linguistique de base utilisé par les applications du système
<code>LC_CTYPE</code>	Le jeu de caractères utilisé pour saisir et afficher du texte
<code>LC_NUMERIC</code>	Mise en forme des valeurs numériques non-monétaires
<code>LC_TIME</code>	Format de la date et de l'heure
<code>LC_COLLATE</code>	Comment trier diverses informations
<code>LC_MONETARY</code>	Format des valeurs numériques monétaires

Expressions régulières

Les ancres

Caractère	Description	Exemple	Que fait l'expression
<code>^</code>	Début de la chaîne	<code>^[0-9]</code>	Affiche uniquement les chaînes qui commencent par un nombre
<code>\$</code>	Fin de la chaîne	<code>[0-9]\$</code>	Affiche uniquement les chaînes qui se terminent par un nombre

Les caractères communs

Caractère	Description	Exemple	Que fait l'expression
<code>.</code>	n'importe quel caractère	<code>[0-9].</code>	Affiche uniquement les chaînes avec un chiffre + un caractère quelconque
<code>*</code>	Une fois ou n fois	<code>[0-9]*</code>	Affiche uniquement les chaînes avec un ou n chiffre
<code>+</code>	1 fois au moins	<code>[0-9]+</code>	Affiche uniquement les chaînes avec au moins un chiffre
<code>?</code>	0 ou 1 fois	<code>[0-9]?</code>	Affiche uniquement les chaînes avec un ou zéro chiffre

Les caractères de classe

Caractère	Equivalent	Description
<code>[abc]</code>		Correspond uniquement au caractère a, b ou c.
<code>[^abc]</code>		Correspond à tous les caractères sauf les caractères a, b et c.
<code>[a-z]</code>		Correspond à tous les caractères entre a-z.
<code>[0-9]</code>		Correspond à tous les caractères entre 0-9.
<code>[a-zA-Z]</code>		Correspond à tous les caractères entre a-z ou A-Z.
<code>[:alnum:]</code>	<code>[A-Za-z0-9]</code>	Correspond à n'importe quelle lettre ou chiffre.
<code>[:alpha:]</code>	<code>[A-Za-z]</code>	Correspond aux lettres de l'alphabet.
<code>[:ascii:]</code>	<code>[\x00-\x7F]</code>	Correspond à n'importe quel caractère de la plage ASCII valide.
<code>[:blank:]</code>	<code>[\t]</code>	Correspond aux espaces et aux tabulations.
<code>[:digit:]</code>	<code>\d</code> ou <code>[0-9]</code>	Correspond aux chiffres décimaux.

Caractère	Equivalent	Description
<code>[:lower:]</code>	<code>[a-z]</code>	Correspond aux lettres minuscules.
<code>[:space:]</code>	<code>\s</code>	Correspond aux caractères d'espace.
<code>[:upper:]</code>	<code>[A-Z]</code>	Correspond aux lettres majuscules.
<code>[:word:]</code>	<code>\w</code> ou <code>[a-zA-Z0-9_]</code>	Correspond aux lettres, aux chiffres et aux traits de soulignement.
<code>[:xdigit:]</code>	<code>[0-9a-fA-F]</code>	Correspond aux chiffres hexadécimaux, insensible à la casse.

Les quantifieurs

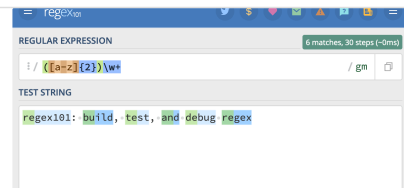
Caractère	Description	Exemple	Que fait l'expression
<code>{n}</code>	Correspond à n caractères consécutifs.	<code>a{3}</code>	Correspond exactement à 3 caractères "a" consécutifs.
<code>{n, }</code>	Correspond à au moins n caractères consécutifs.	<code>a{3, }</code>	Correspond à au moins 3 caractères "a" consécutifs.
<code>{a, b}</code>	Correspond entre a et b caractères consécutifs.	<code>a{3, 6}</code>	Correspond entre 3 et 6 caractères "a" consécutifs.

Liens

regex101: build, test, and debug regex

Regular expression tester with syntax highlighting, explanation, cheat sheet for PHP/PCRE, Python, GO, JavaScript, Java, C#/.NET.

<https://regex101.com/>



Liens utiles

<https://linuxize.com/>