Linux - Cheatsheet

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  # Changer son propre mot de passe
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
```

Les bases

Le système des fichiers et les droits standards

```
# Syntaxe
ls [ OPTIONS ] # Liste les fichiers «ordinaires» 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 l'ordre de tri (rev.)
```

```
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 rep.
cp [-r] source destination    # copie de fichiers
cp -r rép_orig rép_dest       # Copier des répertoires entiers (rec.)
```

```
rsync a rép_orig/ rép_dest/   # Se comporte comme la précédente
mv fichier_orig fichier_dest  # Renommer un fichier, lien ou
répertoire

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

# Afficher les méta-données du fichier
stat fichier
# "Création" d'un fichier fic - modification de l'horodatage
touch fic
# Affiche les diff. entre les deux fichiers
diff fic_a.txt fic_b.txt
# Afficher répertoire courant (print working dir)
pwd
```

Lien Symbolique et Physique

```
# Syntaxe d'un lien "Symbolique" (identique à un raccourcis)
#  Fichier qui « pointe » vers un autre fichier (cible)
#  Si le fichier cible est supprimé le lien symbolique est cassé !
ln -s cible lien

# Exemple
#  1] Copie du fichier hosts dans le rep. courant en hosts1
cp /etc/hosts ~/hosts1
#  2] Création du lien vers hosts1, qui se nomme hosts1.lnk
ln -s hosts1 hosts1.lnk
#  Le lien symbolique est représenter par une flèche "->"
#  vers le fichier qu'il pointe
```

La sécurité des fichiers : les droits

Tldr

La commande chmod permet de changer des permissions à l'aide de la notation de texte

```
# Syntaxe :
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 (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 (u, g, o)
chmod u=rwx,g=rwx,o=rwx fic
chmod 777 fic  # Attribut tous les droits sur fic à tous
chmod u+w fic  # Ajouter droits en écriture au propriétaire
chmod g+r fic  # Ajouter droits en lecture au groupe du fichier
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:nouvgroupe rép
```

Numéros d'autorisation de fichier

Permissions	Valeurs	Descriptions
rwx	7	Accorder tous les droits lecture / écriture / exécution

Permissions	Valeurs	Descriptions
`rw-`	6	Accorder uniquement les droits de lecture / écriture
`r-x`	5	Lecture, exécution
`r--`	4	Lecture (read)
`-w-`	2	Ecriture (write)
`--x`	1	Exécution (execute)
`---`	0	Aucun accès

TLDR

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.

La substitution des commandes

TLDR

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 :

```
# 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."

# La substitution de variable
#  Permet de remplacer le nom d'une variable par sa valeur.
echo "Ma maison se trouve à ${MAISON}."

# La substitution de commande avec 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}."

# La substitution de sous-chaîne
#  Permet de remplacer une partie d'une chaîne de caractères par une autre chaîne.
echo "${Bonjour:0:1}onjour"
```

Edition de fichiers

Éditeurs	Exemples
Graphique	gedit, kate / kwrite, geany
Historiques	vi, vim
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

> **Tldr**
>
> La commande `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
```

Nom	Descriptions
UID	Uid (n° d'identifiant) de l'utilisateur

Nom	Descriptions
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:
```

Intéraction 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 processsus 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	Descriptions	Forme 1	Forme 2
1	Instruction (HANG UP) - Fin de session	SIGHUP	
2	Interruption	SIGINT	Ctrl + C
19	Demande de suspension imbloquable	SIGSTOP	Ctrl + Z

Liens utiles

> **Liens utiles**
>
> Liste des signaux

Flux et Filtres

Les flux

Descriptions	Flux	Alternative	Exemples

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

```
# 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
```

Tldr

`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.
```

Tldr

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

```
# Syntaxe
sort [ -r ] [ -o fic_out ] [ -t ca r] [ -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
#    -o : fichier de sortie
```

```
# 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

> Tldr
>
> **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

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

```
# Syntaxe
locate [ OPTIONS ] Fichier

# Options
#    -e : Afficher que les fichiers existants dans le système
#    -c : Compte le nombre d’entrées
```

```
#    -r : Fait une recherche en utilisant les expréssion régulière
#    -i : Ignore la casse
#    -l <Num> : Réduit 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 moin une fois par jour. Il est possible de mettre à jour cette base de données en utilisant la commade : `updatedb`

grep

> **Tldr**
>
> `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

Tldr

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

```
# Syntaxe
whereis [options] commande

# Options
#    -l : Recherche les répertoires dans lesquels la commande whereis
recherche.
#    -p : Recherche uniquement les binaires de commande
```

```
#    -s : Recherche uniquement les fichiers source
#    -m : Recherche 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/binrépertoire
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
#    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

Tldr

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

Les variables d'environnements

Tldr

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
env	Affiche les variables environnements
echo $NAME	Appel une variable d'environnement par son nom
export NAME=value	Créer une variable
$PATH	Affiche le chemin de recherche
$HOME	Affiche le répertoire d'accueil
$SHELL	Le shell courant
export LC_ALL=	Efface le contenu de la variable
unset LCALL	Supprimer complètement une variable d'environnement
$MANPATH	Liste de dossiers où le système doit chercher les pages de manuel.

Quelques variables de paramètres régionaux

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

Expressions régulières

Les ancres

Caractère	Description	Exemple	Que fait l'expression
^	Début de la chaîne	`^[0-9]`	Affiche uniquement les chaînes qui commence par un nombre
$	Fin de la chaîne	`[0-9]$`	Affiche uniquement les chaînes qui se termine par un nombre

Les caractères commun

Caractère	Description	Exemple	Que fait l'expression
.	n'importe quel caractère	[0-9].	Affiche uniquement les chaines avec un chiffre + un caractère quelconque

Caractère	Description	Exemple	Que fait l'expression
*	Une fois ou n fois	[0-9]*	Affiche uniquement les chaines avec un ou n chiffre
+	1 fois au moins	[0-9]+	Affiche uniquement les chaines avec au moins un chiffre
?	0 ou 1 fois	[0-9]?	Affiche uniquement les chaines avec un ou zéro chiffre

Les caractères de classe

Caractère	Equivalent	Description
[abc]	X	Correspond uniquement au caractère a, b ou c.
[^abc]	X	Correspond à tous les caractères sauf les caractères a, b et c.
[a-z]	X	Correspond à tous les caractères entre a-z .
[0-9]	X	Correspond à tous les caractères entre 0-9.
[a-zA-Z]	X	Correspond à tous les caractères entre a-z ou A-Z.
[[:alnum:]]	[A-Za-z0-9]	Correspond à n'importe quelle lettre ou chiffre.
[[:alpha:]]	[A-Za-z]	Correspond aux lettres de l'alphabet.
[[:ascii:]]	[\x00-\x7F]	Correspond à n'importe quel caractère de la plage ASCII valide.
[[:blank:]]	[\t]	Correspond aux espaces et aux tabulations .
[[:digit:]]	\d ou [0-9]	Correspond aux chiffres décimaux.
[[:lower:]]	[a-z]	Correspond aux lettres minuscules.
[[:space:]]	\s	Correspond aux caractères d'espacement.
[[:upper:]]	[A-Z]	Correspond aux lettres majuscules.
[[:word:]]	\w ou [a-zA-Z0-9_]	Correspond aux lettres, aux chiffres et aux traits de soulignement.
[[:xdigit:]]	[0-9a-fA-F]	Correspond aux chiffres hexadécimaux, insensible à la casse.

Les quantifieurs

Caractère	Description	Exemple	Que fait l'expression
{n}	Correspond à n caractères consécutifs.	`a{3}`	Correspond exactement à 3 caractères "a" consécutifs.
{n, }	Correspond à au moins n caractères consécutifs.	`a{3,}`	Correspond à au moins 3 caractères “a” consécutifs.
{a, b}	Correspond entre a et b caractères consécutifs.	`a{3,6}`	Correspond entre 3 et 6 caractères "a" consécutifs.

Liens

---

regex101: build, test, and debug regex