

Memento Shell

Auteur : *Antoine Pernot*

• Gestion des fichiers/répertoires

Créer un répertoire (make directory) :

-p : Créer le répertoire parent si besoin.

```
| mkdir [-p] rep
```

Créer un fichier :

```
| touch fich
```

Changer de répertoire (change dir) :

```
| cd rep
| cd ..      # repertoire parent
| cd        # repertoire personnel
| cd ~alice # repertoire personnel de alice
```

Afficher répertoire courant (print working dir) :

```
| pwd
```

Copier un fichier/répertoire vers un autre :

-r : Copier le répertoire de manière récursive.

```
| cp [-r] orig dest
```

Créer un lien :

-s : Créer un lien symbolique.

```
| ln [-s] orig lien
```

Déplacer/renommer un fichier/répertoire vers un autre (move) :

```
| mv orig dest
```

Supprimer un fichier (remove) :

-r : Supprime un répertoire de manière récursive.

```
| rm [-r] fich
```

Supprimer un répertoire vide (remove directory) :

```
| rmdir rep
```

Lister le contenu d'un répertoire :

-l : Liste détaillée.

-a : Liste tous les fichiers (inclut les fichiers cachés).

-s : Tri par taille.

-t : Tri par date.

-r : Inverse l'ordre de tri.

```
| ls [-latsr] rep
```

• Gestion du contenu des fichiers/des flux Affichage brut de fichiers (non-interactif) :

```
| cat [fich1 [fich2 ...]]
```

Affichage interactif de fichiers :

```
| more [fich1 [fich2 ...]] # sens unique
| less [fich1 [fich2 ...]] # 2 sens
```

Afficher le début d'un fichier :

-n : Affiche n lignes.

```
| head [-n=10] [fich]
```

Afficher la fin d'un fichier :

-n : Affiche n lignes.

```
| tail [-n=10] [fich]
```

Rechercher dans un fichier :

-i : Insensible à la casse.

-v : Affiche les lignes ne correspondant PAS à l'expression.

```
| grep [-iv] expression [fich]
```

Trie les lignes d'un fichier :

-r : Inverse l'ordre.

-R : Trie dans un ordre aléatoire.

```
| sort [-rR] [fich]
```

Compte les éléments d'un fichier :

-l : Compte le nombre de lignes.

-w : Compte le nombre de mots.

-c : Compte le nombre de caractères.

```
| wc [-clw] [fich]
```

Découper un flux de texte :

-d : Délimiteur de découpe.

-f : Sélectionne les champs à renvoyer.

```
| cut [-df] [fich]
```

• Droits d'accès aux fichiers

Changer les droits d'accès aux fichiers :

-R : Change les droits de manière récursive.

```
| chmod [-R] {ugoa}{+-}{rwx} fich
```

Changer le propriétaire du fichier :

-R : Change le propriétaire de manière récursive.

```
| chown [-R] nvuser[:nvgrp] fich
```

Changer le groupe du fichier :

-R : Change le groupe de manière récursive.

```
| chgrp [-R] nvgrp fich
```

• Recherche de fichiers

Rechercher un fichier :

-exec : Exécute une commande en remplaçant {} par le chemin de chacun des fichiers trouvés.

```
| find rep_rech -name regex [-exec cmd {} ';' ]
```

• Gestion des flux de texte

Rediriger la sortie d'une commande vers l'entrée d'une autre :

```
| cat villes.txt | grep Dijon
```

Écrire la sortie d'une commande dans un fichier (écrase le contenu) :

```
| grep Dijon villes.txt > info_Dijon.txt
```

Ajoute la sortie d'une commande à un fichier :

```
| grep Dijon villes.txt >> info_Dijon.txt
```

Ajoute la sortie d'erreurs d'une commande à un fichier :

```
| grep Dijon villes.txt 2>> erreurs_villes.txt
```

• Contrôle de tâches

Affiche les processus en cours d'exécution :

```
| ps aux
| top # Mode interactif
```

Envoie un signal (d'arrêt) au processus :

-9 : Envoie un signal d'arrêt SIGKILL

```
| kill [-9] pid
```

Envoie un signal (d'arrêt) aux processus appelés "nom" :

```
| killall nom
```

• Aide système

Lister les pages de manuel contenant une chaîne de caractère :

```
| apropos chaine
```

Afficher la page de manuel d'une commande :

```
| man cmd
```

Memento scripts

Auteur : *Antoine Pernot*

• Structure de base

```
#!/bin/bash
# Version du script
echo "Bonjour"
exit 0
```

• Les variables

Affecter une variable :

```
message="Coucou!"
read rep # Stocke la reponse utilisateur
```

Appeler une variable :

```
echo $message
```

Appeler un fragment de chaîne de caractères :

```
echo ${message:offset:nbchars}
```

Variables spéciales :

\$*	Contient tous les arguments passés à la fonction.
\$#	Contient le nombre d'argument.
\$?	Contient le code de retour de la dernière opération.
\$0	Contient le nom du script.
\$n	Contient l'argument n.
#!	Contient le PID de la dernière commande lancée.

• Les tableaux

Affectation (1^{ère} méthode) :

```
tab=(valeur1 valeur2 ...)
```

Affectation (2^{ème} méthode) :

```
tab[0]=John Smith
tab[1]=Jane Doe
```

Compter le nombre d'éléments du tableau :

```
len=${#tab[*]}
```

Afficher un élément :

```
echo ${tab[1]}
```

Afficher tous les éléments :

```
echo ${tab[@]}
```

• Les structures de contrôle

Syntaxe :

```
[ -f fichier ]
```

Opérateurs de test :

-e fichier	Contrôle si fichier existe.
-d fichier	Contrôle si fichier existe et est un répertoire.
-f fichier	Contrôle si fichier existe et est un fichier 'normal'.
-w fichier	Contrôle si fichier existe et est en écriture.
-x fichier	Contrôle si fichier existe et est exécutable.
f1 -nt f2	Contrôle si f1 est plus récent que f2.
f1 -ot f2	Contrôle si f1 est plus vieux que f2.

Opérateurs de comparaison numériques :

\$n1 -eq \$n2	Vérifie si les nombres sont égaux. Utiliser = pour les chaînes de caractères.
\$n1 -ne \$n2	Vérifie si les nombres sont différents. Utiliser != pour les chaînes de caractères.
\$n1 -lt \$n2	Vérifie si n1 est inférieur à n2.
\$n1 -le \$n2	Vérifie si n1 est inférieur ou égal à n2.
\$n1 -gt \$n2	Vérifie si n1 est supérieur à n2.
\$n1 -ge \$n2	Vérifie si n1 est supérieur ou égal à n2.

Les opérateurs logiques :

expr1 -a expr2 ou expr1 && expr2	Opérateur ET.
expr1 -o expr2 ou expr1 expr2	Opérateur OU.
! expr	Opérateur NON.

• Les conditions

```
if [ condition1 ]
then
    Condition 1 vraie
elif [ condition2 ]
then
    Condition 2 vraie
else
    Aucune condition vraie
fi
```

• Les boucles

"Tant que ..." :

```
while [ condition ]
do
    Tant que la condition est vraie
done
```

"Pour ..." :

```
for variable in valeurs
do
    instructions
done
```

• Les aiguillages

```
case "$var" in
    val1 | val2 ) Valeur 1 ou 2;;
    val3 ) Valeur 3;;
    * ) Autres valeurs;;
esac
```

• Les fonctions

Déclaration de la fonction :

```
nom_fonction(){
    instructions
}
```

Appel de la fonction :

```
nom_fonction
```

• Les expressions régulières

.	Caractérise n'importe quel caractère.
[abc]	Caractérise n'importe quel caractère inclus entre les crochets.
[^abc]	Caractérise n'importe quel caractère autre que ceux entre les crochets.
^	Caractérise le début de ligne.
\$	Caractérise la fin de ligne.
*	Signifie 0 à n fois le caractère précédent.
+	Signifie 1 à n fois le caractère précédent.
?	Signifie 0 ou 1 fois le caractère précédent.
{N}	Signifie N fois le caractère précédent.
{N,}	Signifie N fois ou plus le caractère précédent.
{N,M}	Signifie entre N et M fois le caractère précédent.
\	Caractère d'échappement.

Memento sed

Auteur : *Antoine Pernot*

• Syntaxe générale

```
sed -e 'programme sed' fichier_a_traiter
sed -f fichier_programme fichier_a_traiter
```

• Exemples

Substitution de la chaîne "toto" par la chaîne "tata" dans tout le fichier :

```
sed 's/toto/tata/g' fich
```

Afficher les trois premières lignes d'un fichier :

```
sed '3q' fich
```

Ne pas afficher les lignes qui contiennent la chaîne "toto" :

```
sed -e '/toto/d' fich
```

Afficher la dernière ligne d'un fichier :

```
sed -n '$p' fich
```

Afficher les lignes 5 à 10 d'un fichier :

```
sed -n '5,10p' fich
```

Afficher les lignes qui commencent par "le" :

```
sed '/^le/p' fich
```

Afficher les lignes qui se terminent par commencent par "filtre" :

```
sed '/filtre$/p' fich
```

Ajouter "ligne:" en début de chaque ligne :

```
sed 's/^/ligne :/' fich
```

Mettre le mot « filtre » entre parenthèses :

```
sed 's/filtre/(&)/g' fich
```

Enregistrer et réutiliser des portions de texte :

```
sed 's/\(.*\).*. \(.*\)/\1:\2/' fich
```

Memento awk

• Syntaxe générale

```
awk [-F<separ>] [-f <fic cmd>] 'PRGM' fic
```

• Syntaxe du programme

Exécuté avant la lecture des données

```
BEGIN {action}
```

Exécuté après la lecture des données

```
END {action}
```

Exécuté pour chaque ligne lue

```
{action}
```

Exécuté pour chaque ligne conforme au modèle

```
modele {action}
```

Exécuté pour chaque ligne entre 1 et 2

```
modele1 , modele2 {action}
```

• Syntaxe du modèle

Expressions régulières

```
/expression/
```

Opérateurs de comparaison :

\$n1 == \$n2	Vérifie si les valeurs sont égales.
\$n1 != \$n2	Vérifie si les valeurs sont inégales.
\$n1 < \$n2	Vérifie si n1 est inférieur à n2.
\$n1 <= \$n2	Vérifie si n1 est inférieur ou égal à n2.
\$n1 > \$n2	Vérifie si n1 est supérieur à n2.
\$n1 >= \$n2	Vérifie si n1 est supérieur ou égal à n2.
\$n1 ~ /exp/	Vérifie si n1 contient l'expression.
\$n1 !~ /exp/	Vérifie si n1 ne contient pas l'expression.
&&	ET logique entre deux modèles.
	OU logique entre deux modèles.

Fonctions :

a=1	Affectation.
a+=1	Incrémement.
if (exp) action else action	Condition.
while (exp) action	Boucle Tant que.
do action while (exp)	Boucle Faire.
for (init; cond; incr) action	Boucle Pour.
print (exp)	Imprimer.

Les variables prédéfinies :

ARGC	Nombre d'arguments de la ligne de commande.
ARGV	Tableau des arguments de la ligne de commande.
FILENAME	Nom du fichier sur lequel on applique les commandes.
FNR	Nombre d'enregistrements du fichier.
FS	Séparateur de champs en entrée.
NF	Nombre de champs de l'enregistrement courant.
NR	Nombre d'enregistrements déjà lu.
OFMT	Format de sortie des nombres.
OFS	Séparateur de champs pour la sortie.
ORS	Séparateur d'enregistrement pour la sortie.
RLENGTH	Longueur de la chaîne trouvée.
RS	Séparateur d'enregistrement en entrée.
RSTART	Début de la chaîne trouvée.
SUBSEP	Séparateur de subscript.

Fonctions sur les chaînes de caractères :

gsub(r,s,t)	Sur la chaîne t, remplace toutes les occurrences de r par s
index(s,t)	Retourne la position la plus à gauche de la chaîne t dans la chaîne s
length(s)	Retourne la longueur de la chaîne s
match(s,r)	Retourne l'index où s correspond à r et positionne RSTART et RLENGTH
split(s,a,fs)	Split s dans le tableau a sur fs, retourne le nombre de champs
sub(r,s,t)	Comme gsub, mais remplace uniquement la première occurrence
substr(s,i,n)	Retourne la sous chaîne de s commençant en i et de taille n

Fonctions numériques :

atan2(y,x)	Arc-tangente de $\frac{x}{y}$ en radians dans l'intervalle $-\pi \pi$
cos(x)	Cosinus (en radians)
exp(x)	Exponentielle e à la puissance x
int(x)	Valeur entière
log(x)	Logarithme naturel
rand()	Nombre aléatoire entre 0 et 1
sin(x)	Sinus (en radians)
sqrt(x)	Racine carrée
srand(x)	Réinitialiser le générateur aléatoire