Dernière mise-à-jour : 2020/01/30 03:28

SO102 - Outils de Manipulation de Fichiers Texte

Expressions Régulières

La manipulation de fichiers textes utilise des **expressions régulières**. Sous Linux il existe deux types d'expressions régulières :

- expressions régulières basiques, appelées **ERb**,
  - utilisées par les commandes **vi**, **grep**, **expr** et **sed**,
- expressions régulières étendues, appelées **ERe**,
  - utilisées par les commandes **egrep** (grep -E) et **awk**.

Les expressions régulières utilisent des caractères spéciaux. Certains caractères sont communs aux Erb et aux Ere :

Caractère spécial	Description
^	Trouver la chaîne au début de la ligne
$	Trouver la chaîne à la fin de la ligne
\	Annuler l'effet spécial du caractère suivant
[ ]	Trouver n'importe quel des caractères entre les crochets
[^]	Exclure les caractères entre crochets
.	Trouver n'importe quel caractère sauf à la fin de la ligne
*	Trouver 0 ou plus du caractère qui précède
\<	Trouver la chaîne au début d'un mot
\>	Trouver la chaîne à la fin d'un mot

ERb

Certains caractères spéciaux sont spécifiques aux ERb :

Caractère spécial	Description
\{a,b\}	Trouver de **a** à **b** occurrences de ce qui précède
\{a\}	Trouver exactement le nombre **a** d'occurrences de ce qui précède
\{a,\}	Trouver le nombre **a** ou plus d'occurrences de ce qui précède
\(ERb)	Mémoriser une ERb
\1	Rappeler la première ERb mémorisée
\2, \3 …	Rappeler la deuxième ERb mémorisée, rappeler la troisième ERb mémorisée etc

ERe

Certains caractères spéciaux sont spécifiques aux ERe :

Caractère spécial	Description
?	Trouver 0 ou 1 occurrence de ce qui précède
+	Trouver 1 ou **n** d'occurrences de ce qui précède
{a,b}	Trouver de **a** à **b** occurrences de ce qui précède
{a}	Trouver exactement le nombre **a** d'occurrences de ce qui précède
{a,}	Trouver le nombre **a** ou plus d'occurrences de ce qui précède
()	Faire un **ET** des expressions régulières entre les paranthèses
|	Faire un **OU** des expressions régulières se trouvant de chaque côté du pipe

La Commande grep

La commande grep peut être utilisée pour rechercher des lignes contenant une chaîne de caractères dans un jeu de fichiers.

Par défaut, la commande grep est sensible à la casse. Pour rendre cette commande insensible à la casse, il faut utiliser l'option **-i**.

La commande grep peut être aussi utilisée pour faire l'inverse, autrement dit de montrer les lignes qui ne contiennent pas la chaîne recherchée. Dans ce cas, il faut utiliser l'option **–v**.

La commande grep peut être utilisée avec des **Expressions Régulières basiques**. Ceci est utile pour rechercher dans le contenu de fichiers.

<note> Téléchargez le fichier **greptest** de la section **Fichiers partagés** de cette leçon et placez-le dans le répertoire **/tmp**. </note>

Le contenu du fichier greptest est :

greptest

```
fenestrOS
fenestros
555-5555
f
.fenestros
.fe
£
```

Recherchez maintenant toute ligne du fichier **/tmp/greptest** contenant au moins une lettre :

```
# grep '[a-zA-Z]' /tmp/greptest
fenestrOS
fenestros
f
.fenestros
.fe
```

Recherchez maintenant toute ligne contenant au moins une lettre ou un chiffre :

```
# grep '[a-zA-Z0-9]' /tmp/greptest
fenestrOS
fenestros
555-5555
f
.fenestros
.fe
```

<note important> Notez la présence de la ligne **555-5555**. </note>

Recherchez maintenant toute ligne contenant un numéro de téléphone au format NNN-NNNN :

```
# grep '[0-9]\{3\}-[0-9]\{4\}' /tmp/greptest
555-5555
```

Recherchez maintenant toute ligne contenant exactement un caractère :

```
# grep '^.$' /tmp/greptest
f
£
```

<note important> Notez l'utilisation des caractères spéciaux le **début de ligne** : **^**, **n'importe quel caractère** : **.** et la **fin de ligne** : **$**. </note>

Recherchez maintenant toute ligne commençant par un point :

```
# grep '^\.' /tmp/greptest
.fenestros
.fe
```

<note important> Notez l'utilisation du caractère d'échappement **** pour annuler l'effet du caractère spécial **.** </note>

La Commande egrep

Avec la commande **egrep**, l'utilisation des expressions régulières est étendue aux ERe.

Éditez maintenant votre fichier **/tmp/greptest** ainsi :

greptest1

```
# Commentaire du début
```

```
fenestrOS
fenestros
# Un autre commentaire
555-5555
f

.fenestros

.fe

£
# Commentaire de la fin
```

Utilisez la commande **egrep** pour envoyer le contenu du fichier **/tmp/greptest**, sans commentaires et sans lignes vides, dans le fichier **/tmp/greptest1** :

```
[root@centos ~]# egrep -v '^(#|$)'  /tmp/greptest > /tmp/greptest1
[root@centos ~]# cat /tmp/greptest1
fenestrOS
fenestros
555-5555
f
.fenestros
.fe
£
```

<note important> Cette commande est particulièrement utile face à un fichier de configuration de plusieurs centaines de lignes dont certaines contiennent des directives activées d'autres sont vides ou en commentaires. De cette façon vous pouvez généré facilement un fichier ne contenant que les directives activées. </note>

La Commande fgrep

Avec la commande **fgrep**, la recherche concerne une chaîne de caractères interprétés dans un sens littéral sans utilisation de caractères spéciaux ni d'expressions régulières.

Éditez maintenant votre fichier **/tmp/greptest** ainsi :

greptest2

```
# Commentaire du début
^ voici une ligne pour la recherche fgrep
fenestrOS
fenestros
# Un autre commentaire
555-5555
f

.fenestros

.fe

£
# Commentaire de la fin
```

Utilisez maintenant la commande **fgrep** pour rechercher la ligne commençant par le caractère **^** :

```
# fgrep '^' /tmp/greptest
^ voici une ligne pour la recherche fgrep
```

Comparez le résultat ci-dessus avec celui de la commande grep :

```
# grep '^' /tmp/greptest
```

```
# Commentaire du début
^ voici une ligne pour la recherche fgrep
fenestrOS
fenestros
# Un autre commentaire
555-5555
f

.fenestros

.fe

£
# Commentaire de la fin
```

En effet, la ligne de commande en utilisant la commande grep devrait être :

```
# grep '^\^' /tmp/greptest
^ voici une ligne pour la recherche fgrep
```

Le Commande sed

La commande **sed** ou *Stream EDitor* est un éditeur de texte non-intéractif. Les actions spécifiées par la commande sed sont exécutées par défaut sur chaque ligne du fichier. La commande sed ne modifie pas le fichier d'origine et sa sortie standard est le canal 1.

Si plusieurs actions sont spécifiées dans la ligne de commande, chacune doit être précédée par l'option **-e**.

La syntaxe de la commande sed est la suivante :

```
sed [adresse] commande [arguments]
```

L'**adresse** permet de stipuler les lignes concernées par la **commande**.

La syntaxe d'une adresse peut être :

adresse	Lignes concernées
a	La ligne numéro **a**
$	La dernière ligne
/ERb/	Les lignes qui correspondent à l'ERb
a,b	De la ligne numéro **a** jusqu'à la ligne numéro **b**
/ERb1/, /ERb2/	Toutes les lignes entre la première occurrence correspondant à l'ERb1 jusqu'à la première occurrence correspondant à l'ERb2

Le commandes de sed sont :

commande	Description
d	Ne pas afficher la ou les ligne(s)
p	Afficher la ou les ligne(s)
s	Effectuer une substitution
w	Ecrire le ou les ligne(s) dans un fichier
=	Afficher le numéro de la ligne spécifiée
!	Exécuter la commande ci-dessus sur toutes les lignes sauf celle spécifiées dans l'adresse

La Commande d

La commande **d** de sed permet de ne pas afficher certaines lignes à l'écran. Dans l'exemple qui suit, les 10 premières lignes du fichier **/etc/services** ne sont pas affichées à l'écran :

```
# sed '1,10d' /etc/services | more
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
chargen         19/tcp          ttytst source
```

```
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp             21/tcp
ssh             22/tcp                          # Secure Shell
telnet          23/tcp
smtp            25/tcp          mail
time            37/tcp          timserver
time            37/udp          timserver
name            42/udp          nameserver
whois           43/tcp          nicname         # usually to sri-nic
domain          53/udp
domain          53/tcp
bootps          67/udp                          # BOOTP/DHCP server
bootpc          68/udp                          # BOOTP/DHCP client
--A suivre--
```

Dans l'exemple qui suit, sed n'affiche pas de lignes de commentaires, c'est-à-dire les lignes commençant par le caractère **#** :

```
# sed '/^#/d' /etc/services | more
tcpmux          1/tcp
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp             21/tcp
ssh             22/tcp                          # Secure Shell
telnet          23/tcp
```

```
smtp            25/tcp          mail
time            37/tcp          timserver
time            37/udp          timserver
name            42/udp          nameserver
whois           43/tcp          nicname         # usually to sri-nic
domain          53/udp
domain          53/tcp
--A suivre--
```

<note important> Notez que l'ERb est entourée des caractères / et /. </note>

La Commande p

La commande sed vous permet d'afficher à l'écran certaines lignes spécifiées en utilisant la commande **p** :

```
# sed '1,2p' /etc/passwd
root:x:0:0:Super-User:/:/sbin/sh
root:x:0:0:Super-User:/:/sbin/sh
daemon:x:1:1::/:
daemon:x:1:1::/:
bin:x:2:2::/usr/bin:
sys:x:3:3::/:
adm:x:4:4:Admin:/var/adm:
lp:x:71:8:Line Printer Admin:/usr/spool/lp:
uucp:x:5:5:uucp Admin:/usr/lib/uucp:
nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/uucico
smmsp:x:25:25:SendMail Message Submission Program:/:
listen:x:37:4:Network Admin:/usr/net/nls:
gdm:x:50:50:GDM Reserved UID:/:
webservd:x:80:80:WebServer Reserved UID:/:
postgres:x:90:90:PostgreSQL Reserved UID:/:/usr/bin/pfksh
svctag:x:95:12:Service Tag UID:/:
nobody:x:60001:60001:NFS Anonymous Access User:/:
```

```
noaccess:x:60002:60002:No Access User:/:
nobody4:x:65534:65534:SunOS 4.x NFS Anonymous Access User:/:
```

<note important> Notez que sed affiche également tout le contenu du fichier. Ceci implique que les lignes 1 et 2 s'affichent deux fois. </note>

Pour n'afficher que les lignes spécifiées, il convient d'utiliser l'option **-n** :

```
# sed -n '1,2p' /etc/passwd
root:x:0:0:Super-User:/:/sbin/sh
daemon:x:1:1::/:
```

La Commande w

La commande **w** permet d'écrire dans un fichier. Par exemple pour écrire dans le fichier **/tmp/sedtest** toutes les lignes du fichier **/etc/services** ne commençant pas par le caractère **#**, il convient d'utiliser la commande suivante :

```
# sed -n '/^#/!w /tmp/sedtest' /etc/services
# more /tmp/sedtest
tcpmux          1/tcp
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp             21/tcp
ssh             22/tcp                          # Secure Shell
telnet          23/tcp
```

```
smtp            25/tcp          mail
time            37/tcp          timserver
time            37/udp          timserver
name            42/udp          nameserver
whois           43/tcp          nicname         # usually to sri-nic
domain          53/udp
domain          53/tcp
--A suivre--(11%)
```

La Commande s

La commande **s** permet de procéder à une substitution :

```
# echo "user1,user2,user3" > /tmp/sedtest1
# cat /tmp/sedtest1
user1,user2,user3
# cat /tmp/sedtest1 | sed 's/,/ /g'
user1 user2 user3
```

<note important> Notez que dans cet exemple, la commande **s** est suivi par un argument qui prend la forme **/ce qui est à remplacer (caractère, chaîne ou ERb)/chaîne de remplacement/g**. Le caractère **g** force le remplacement de toutes les occurrences. Sans elle, uniquement la première occurrence serait remplacée. Dans le cas de l'exemple, on remplace donc les virgules par des espaces. </note>

La Commande awk

Présentation

Le processeur de texte **awk** est un **filtre**. Une **action** awk est fournie sur la ligne de commande entourée de ' ou de “ :

```
awk [-F séparateur] 'critère {action}' [fichier1 ... fichiern]
```

<note important> Le couple **critère {action}** s'appelle une **clause**. </note>

Dans le cas de l'utilisation d'un **script** awk, la syntaxe de la commande devient :

```
awk [-F séparateur] -f script [fichier1 ... fichiern]
```

Découpage en champs

awk sait identifier les champs de la ligne soit parce que ceux-ci sont séparés par un espace ou par une tabulation soit parce que la ligne de commande lui a identifié le séparateur grâce à l'option **-F**.

awk stocke les informations de la ligne dans des variables :

Variable	Description
$0	Contient toute la ligne
$1, $2 ...	Contient le premier champ de la ligne, contient le deuxième champ de la ligne ...

Par exemple :

```
# pwd
/Documents
# ls -l | awk '{print $8 $3 $4}'

15:54rootroot
15:54rootroot
15:54rootroot
15:54rootroot
```

Comme vous pouvez constater, awk a extrait du résultat de la commande **ls-l** les champs **nom de l'élément**, **le propriétaire** et le **groupe**.

Afin de le rendre un peu plus lisible, saisissez la commande suivante :

```
# pwd
```

```
/Documents
# ls -l | awk '{print $8 " " $3 " " $4}'

15:54 root root
15:54 root root
15:54 root root
15:54 root root
```

Critères

Les **critères** conditionnent l'exécution d'une **action** dans une **clause**.

Plusieurs types de critères sont possibles. Les plus utilisées sont les suivantes :

Une expression régulière valide pour la ligne

- Format:
- /expression régulière/ {instruction}
- Exemple:
- /ERe/ {print $0}

Une expression régulière valide pour un champ

- Format:
- $n ~/expression régulière/ {instruction}
- $n!~/expression régulière/ {instruction}
- Exemple:
- $1 ~/ERe/ {print $0}
- $1!~/ERe/ {print $0}

awk sélectionne des lignes en utilisant un opérateur de correspondance ou de non-correspondance :

Opérateur	Condition
~	Correspondance
!~	Non-correspondance

Une comparaison

- Format:
- $n opérateur critère de comparaison {action}
- Exemple:
- $1 > 20 {print $0}

Les opérateurs sont :

Opérateur	Condition
<	Inférieur
<=	Inférieur ou égal
==	Egal
!=	Différent
>	Supérieur
>=	Supérieur ou égal

Un opérateur logique

- Format:
- test1 opérateur logique test2 {action}
- Exemple:
- $1 ~/ERe/ && $2 > 20 {print $0}

Les opérateurs sont :

Opérateur logique	Condition
||	OU
&&	ET

Opérateur logique	Condition
!	NON

Une variable interne

- Format:
- expression1, expression2 {action}
- Exemple:
- NR==7, NR==10 {print $0}

Les variables sont :

Variable	Description
NR	Nombre total de lignes
NF	Nombre total de champs
FILENAME	Le nom du fichier en entrée
FS	Le séparateur de champs en entrée. Par défaut un **espace** ou une **tabulation**
RS	Le séparateur de lignes en entrée. Par défaut une **nouvelle ligne**
OFS	Le séparateur de champs en sortie. Par défaut un **espace**
ORS	Le séparateur de lignes en sortie. Par défaut une **nouvelle ligne**
OFMT	Le format numérique. Par défaut ”%.6g“

Scripts awk

Quand un programme awk comporte plusieurs **clauses** composées de **critères** et d'**actions**, il convient de d'écrire un **script awk**. Ce script comporte trois sections :

<note important>

- La section **BEGIN**
  - Cette section est exécutée avant la lecture du script
- La section **principale**
  - Cette section contient les clauses

- La section **END**
  - Cette sectione est exécutée une fois à la fin du script

</note>

Par exemple :

```
[root@centos ~]# cd /tmp
[root@centos tmp]# cat > scriptawk
BEGIN {
  print "Liste des systèmes de fichiers montés"}
{print $0}
END {
  print "=====================================" [Entrée]
[^D]
```

<note important> Dans l'exemple ci-dessus, la ligne **[^D]** indique que vous devez appuyer simultanément sur les touches CTRL et D. </note>

Ensuite saisissez la commande suivante :

```
# awk -f scriptawk /etc/vfstab
Liste des systèmes de fichiers montés
#device         device          mount           FS      fsck    mount   mount
#to mount       to fsck         point           type    pass    at boot options
#
fd      -       /dev/fd fd      -       no      -
/proc   -       /proc   proc    -       no      -
/dev/dsk/c0d0s1 -       -       swap    -       no      -
/dev/dsk/c0d0s0 /dev/rdsk/c0d0s0        /       ufs     1       no      -
/dev/dsk/c0d0s7 /dev/rdsk/c0d0s7        /export/home    ufs     2       yes    -/devices        -       /devices
devfs   -       no      -
sharefs -       /etc/dfs/sharetab       sharefs -       no      -
ctfs    -       /system/contract        ctfs    -       no      -
objfs   -       /system/object  objfs   -       no      -
```

```
swap    -       /tmp    tmpfs   -       yes     -
=====================================
```

<note important> Notez l'utilisation de l'option **-f** qui applique le script awk au fichier donné en argument. </note>

La Fonction printf

La fonction intégrée **printf** permet de formater des affichages. Elle a la syntaxe suivante :

```
printf ("chaine",expression1,expression2,...,expressionn)
```

chaine contient autant de formats qu'il y a d'expressions.

Les formats de printf sont, par exemple :

Format	Description
%30s	Affichage d'une chaîne (s=string) sur 30 positions avec cadrage à droite
%-30s	Affichage d'une chaîne (s=string) sur 30 positions avec cadrage à gauche
%4d	Affichage d'un entier sur 4 positions avec cadrage à droite
%-4d	Affichage d'un entier sur 4 positions avec cadrage à gauche

Structures de Contrôle

awk peut utiliser des structures de contrôle.

if

La syntaxe de la commande if est la suivante :

```
if condition {
```

```
    commande
    commande
    ...
}

else {

    commande
    commande
    ...
}
```

ou dans le cas d'une seule commande :

```
if condition

    commande

else

    commande
```

for

La syntaxe de la structure de contrôle **for** est la suivante :

```
for variable in liste_variables {

    commande
    commande
    ...
```

```
}
```

ou dans le cas d'une seule commande :

```
for variable in liste_variables

    commande
```

ou dans le cas d'un tableau :

```
for clef dans tableau {

    print clef , tableau[clef]

}
```

while

La syntaxe de la structure de contrôle **while** est la suivante :

```
while condition {

     commande
     commande
     ...

}
```

do-while

La syntaxe de la structure de contrôle **do-while** est la suivante :

```
do {

     commande
     commande
     ...

} while condition
```

Tableaux

Pour illustrer l'utilisation des tableaux, créez le fichier **ventes_materiel**. Ce fichier contient des statistiques de vente par type de PC et par département :

ventes_materiel

```
# FenestrOs.com
# Ventes annuelles par département
# 83
Desktops§100
Portables§50
Serveurs§21
Tablettes§4

# 06
Desktops§99
Portables§60
Serveurs§8
Tablettes§16

# 13
Desktops§130
Portables§65
```

```
Serveurs§12
Tablettes§56
```

Créez maintenant le script awk **ventes.awk**. Ce script a pour but de calculer le nombre total de PC vendus dans les trois départements cités dans le fichier **ventes_materiel** :

ventes.awk

```
# BEGIN
BEGIN {
        FS="§"
        }
# TABLEAU
$1 !~ /^#/ && $1 !~ /^$/ {
        ventes[$1]+=$2
}
# END
END {
        for (pc in ventes)
        printf("Type PC :  %s \t Ventes (06+13+83) : %10d\n",pc,ventes[pc]);
}
```

Ce fichier comporte 13 lignes :

```
 1  # BEGIN
 2  BEGIN {
 3      FS="§"
 4  }
 5  # TABLEAU
 6  $1 !~ /^#/ && $1 !~ /^$/ {
 7      ventes[$1]+=$2
 8  }
```

```
 9  # END
10  END {
11      for (pc in ventes)
12      printf("Type PC :  %s \t Ventes (06+13+83) : %10d\n",pc,ventes[pc]);
13  }
```

Dans ce script vous noterez :

- La ligne **3**,
  - Cette ligne se trouve dans la section **BEGIN**. Elle spécifie le séparateur de champs.
- La ligne **6**,
  - Cette ligne évite le traitement de toute ligne commençant par le caractère **#** ainsi que toute ligne vide.
- La ligne **7**,
  - Ce tableau a pour clef la valeur de **$1**, c'est-à-dire, les noms des différents types de PC. Le valeurs du tableau sont le nombre de PC vendus, ici représenté par **$2**. Les caractères **+=** indique qu'à chaque traitement de ligne, le nombre de PC vendus sur la ligne doit être rajouté à la valeur déjà présente dans le tableau.
- La ligne **11**,
  - Cette ligne démarre une boucle **for**.7
- La ligne **12**,
  - Cette ligne utilise **printf** afin d'imprimer à l'écran les valeurs calculées et stockées dans le tableau.

Appliquez maintenant votre script awk au fichier **ventes_materiel** :

```
# awk -f ventes.awk ventes_materiel
Type PC :  Serveurs      Ventes (06+13+83) :         41
Type PC :  Tablettes     Ventes (06+13+83) :         76
Type PC :  Portables     Ventes (06+13+83) :        175
Type PC :  Desktops      Ventes (06+13+83) :        329
```

La Commande cut

Chaque ligne est divisée en colonnes. Dans une ligne le premier caractère est dans la colonne numéro **un**, le deuxième dans la colonne deux et ainsi

de suite. Dans une ligne il peut y avoir des champs séparés par des tabulations.

La commande **cut** permet de sélectionner des colonnes et des champs dans un fichier. La commande permet aussi d'utiliser une critère de séparation de champs autre que la tabulation en spécifiant cette critère en utilisant l'option **-d**.

Par exemple, pour sélectionner les 7 premières colonnes du fichier **/etc/passwd** la commande est :

```
# cut -c1-7 /etc/passwd
root:x:
daemon:
bin:x:2
sys:x:3
adm:x:4
lp:x:71
uucp:x:
nuucp:x
smmsp:x
listen:
gdm:x:5
webserv
postgre
svctag:
nobody:
noacces
nobody4
test:x:
```

Pour sélectionner les colonnes 1 à 5, les colonnes 10 à 15 et les colonnes 30 et après, il convient d’utiliser la commande suivante :

```
# cut -c1-5,10-15,30- /etc/passwd
root:0:Supe/sh
daemo1:1::/
bin:x::/usr
sys:x::/:
```

```
adm:x:Admin
lp:x::Line /usr/spool/lp:
uucp:5:uucpb/uucp:
nuucp:9:uucpool/uucppublic:/usr/lib/uucp/uucico
smmsp5:25:Se Submission Program:/:
liste37:4:Nusr/net/nls:
gdm:x50:GDM/:
websex:80:8served UID:/:
postgx:90:9eserved UID:/:/usr/bin/pfksh
svcta95:12:D:/:
nobod60001:ymous Access User:/:
noaccx:6000ess User:/:
nobod:65534.x NFS Anonymous Access User:/:
test:0:1:Tee/test:/bin/ksh
```

Pour sélectionner les champs 2, 4 et 6 du fichier, il convient d'utiliser la commande suivante :

```
# cut -d: -f2,4,6 /etc/passwd
x:0:/
x:1:/
x:2:/usr/bin
x:3:/
x:4:/var/adm
x:8:/usr/spool/lp
x:5:/usr/lib/uucp
x:9:/var/spool/uucppublic
x:25:/
x:4:/usr/net/nls
x:50:/
x:80:/
x:90:/
x:12:/
x:60001:/
x:60002:/
```

```
x:65534:/
x:1:/export/home/test
```

La Commande uniq

La commande suivante permet d'extraire du fichier /etc/passwd les GID utilisés en tant que groupes principaux des utilisateurs :

```
# cut -d: -f4 /etc/passwd | sort -n | uniq
0
1
2
3
4
5
8
9
12
25
50
80
90
60001
60002
65534
```

<note important> Notez l'utilisation de la commande **uniq** qui permet de supprimer les doublons dans la sortie triée. </note>

La Commande tr

La commande **tr** permet de substituer des caractères pour d'autres. Cette commande n'accepte que des données en provenance de son entrée

standard et non en provenance d'un fichier.

```
# cat /etc/passwd | tr "[a-z]" "[A-Z]"
ROOT:X:0:0:SUPER-USER:/:/SBIN/SH
DAEMON:X:1:1::/:
BIN:X:2:2::/USR/BIN:
SYS:X:3:3::/:
ADM:X:4:4:ADMIN:/VAR/ADM:
LP:X:71:8:LINE PRINTER ADMIN:/USR/SPOOL/LP:
UUCP:X:5:5:UUCP ADMIN:/USR/LIB/UUCP:
NUUCP:X:9:9:UUCP ADMIN:/VAR/SPOOL/UUCPPUBLIC:/USR/LIB/UUCP/UUCICO
SMMSP:X:25:25:SENDMAIL MESSAGE SUBMISSION PROGRAM:/:
LISTEN:X:37:4:NETWORK ADMIN:/USR/NET/NLS:
GDM:X:50:50:GDM RESERVED UID:/:
WEBSERVD:X:80:80:WEBSERVER RESERVED UID:/:
POSTGRES:X:90:90:POSTGRESQL RESERVED UID:/:/USR/BIN/PFKSH
SVCTAG:X:95:12:SERVICE TAG UID:/:
NOBODY:X:60001:60001:NFS ANONYMOUS ACCESS USER:/:
NOACCESS:X:60002:60002:NO ACCESS USER:/:
NOBODY4:X:65534:65534:SUNOS 4.X NFS ANONYMOUS ACCESS USER:/:
TEST:X:100:1:TEST:/EXPORT/HOME/TEST:/BIN/KSH
```

La commande tr peut être utilisée pour isoler une chaîne. Par exemple pour isoler l'adresse IP de la carte eth0 dans la sortie de la commande ifconfig :

```
# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
e1000g0: flags=1004843<UP,BROADCAST,RUNNING,MULTICAST,DHCP,IPv4> mtu 1500 index 2
        inet 10.0.2.15 netmask ffffff00 broadcast 10.0.2.255
        ether 8:0:27:42:1d:38
# ifconfig e1000g0
e1000g0: flags=1004843<UP,BROADCAST,RUNNING,MULTICAST,DHCP,IPv4> mtu 1500 index 2
        inet 10.0.2.15 netmask ffffff00 broadcast 10.0.2.255
        ether 8:0:27:42:1d:38
```

```
# ifconfig e1000g0 | grep "inet"
        inet 10.0.2.15 netmask ffffff00 broadcast 10.0.2.255
# ifconfig e1000g0 | grep "inet" | tr -s " " ":"
        inet:10.0.2.15:netmask:ffffff00:broadcast:10.0.2.255
# ifconfig e1000g0 | grep "inet" | tr -s " " ":" | cut -d: -f2
10.0.2.15
```

<note important> Notez l'utilisation de l'option **-s** avec la commande tr. Cette option permet de remplacer une suite de x caractères identiques par un seul caractère. </note>

La Commande paste

La commande **paste** concatène les lignes de n fichiers. Par exemple :

```
# paste -d: /etc/passwd /etc/shadow
root:x:0:0:Super-User:/:/sbin/sh:root:D24f5zPurB8qI:6445::::::
daemon:x:1:1::/::daemon:NP:6445::::::
bin:x:2:2::/usr/bin::bin:NP:6445::::::
sys:x:3:3::/::sys:NP:6445::::::
adm:x:4:4:Admin:/var/adm::adm:NP:6445::::::
lp:x:71:8:Line Printer Admin:/usr/spool/lp::lp:NP:6445::::::
uucp:x:5:5:uucp Admin:/usr/lib/uucp::uucp:NP:6445::::::
nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/uucico:nuucp:NP:6445::::::
smmsp:x:25:25:SendMail Message Submission Program:/::smmsp:NP:6445::::::
listen:x:37:4:Network Admin:/usr/net/nls::listen:*LK*:::::::
gdm:x:50:50:GDM Reserved UID:/::gdm:*LK*:::::::
webservd:x:80:80:WebServer Reserved UID:/::webservd:*LK*:::::::
postgres:x:90:90:PostgreSQL Reserved UID:/:/usr/bin/pfksh:postgres:NP:::::::
svctag:x:95:12:Service Tag UID:/::svctag:*LK*:6445::::::
nobody:x:60001:60001:NFS Anonymous Access User:/::nobody:*LK*:6445::::::
noaccess:x:60002:60002:No Access User:/::noaccess:*LK*:6445::::::
nobody4:x:65534:65534:SunOS 4.x NFS Anonymous Access User:/::nobody4:*LK*:6445::::::
```

```
test:x:100:1:Test:/export/home/test:/bin/ksh:test:kFNaFtSX26m4Q:15568::::::
```

La Commande split

La commande **split** est utilisée pour découper de grands fichiers en petit morceaux d'un taille fixe ou d'un nombre de lignes fixe.

Créez d'abord un fichier d'une taille de 250Mo :

```
# dd if=/dev/zero of=/file bs=1024k count=250
250+0 enregistrements entrants
250+0 enregistrements sortants
```

Utilisez maintenant la commande **split** pour diviser ce fichier en morceaux de 50 Mo :

```
# split -b 50m /file filepart
# ls -l | grep filepart
-rw-r--r--   1 root     root     52428800 août 16 12:00 filepartaa
-rw-r--r--   1 root     root     52428800 août 16 12:00 filepartab
-rw-r--r--   1 root     root     52428800 août 16 12:00 filepartac
-rw-r--r--   1 root     root     52428800 août 16 12:00 filepartad
-rw-r--r--   1 root     root     52428800 août 16 12:00 filepartae
```

<note important> Notez que cinq morceaux ont été créés dans le répertoire courant. </note>

La Commande diff

La commande **diff** indique les modifications à apporter à deux fichiers pour que ceux-ci soient identique.

Pour commencer, créez l'utilisateur **test** :

```
# useradd -d /export/home/test -m -s /bin/ksh -c "Test" test
64 blocs
```

Puis utilisez la commande **passwd** pour créer son mot de passe :

```
# passwd test
Nouveau mot de passe : test
Entrez de nouveau le mot de passe : test
passwd: mot de passe correctement modifié pour test
```

<note important> Notez que le mot de passe saisi ne sera **pas** visible. </note>

Copiez le fichier **/etc/passwd** vers le répertoire **/** :

```
[root@centos ~]# cp /etc/passwd /
```

Modifiez ensuite le fichier la ligne **test** du fichier /passwd ainsi :

```
...
test10:x:500:500:trainee:/home/trainee:/bin/bash
...
```

Supprimez les lignes **nuucp** et **svctag** dans le fichier /passwd et ajoutez en fin de fichier la ligne **Unix est super!** :

```
# vi passwd
"passwd" 18 lignes, 719 caractères
root:x:0:0:Super-User:/:/sbin/sh
daemon:x:1:1::/:
bin:x:2:2::/usr/bin:
sys:x:3:3::/:
adm:x:4:4:Admin:/var/adm:
lp:x:71:8:Line Printer Admin:/usr/spool/lp:
uucp:x:5:5:uucp Admin:/usr/lib/uucp:
```

```
smmsp:x:25:25:SendMail Message Submission Program:/:
listen:x:37:4:Network Admin:/usr/net/nls:
gdm:x:50:50:GDM Reserved UID:/:
webservd:x:80:80:WebServer Reserved UID:/:
postgres:x:90:90:PostgreSQL Reserved UID:/:/usr/bin/pfksh
nobody:x:60001:60001:NFS Anonymous Access User:/:
noaccess:x:60002:60002:No Access User:/:
nobody4:x:65534:65534:SunOS 4.x NFS Anonymous Access User:/:
test10:x:100:1:Test:/export/home/test:/bin/ksh
Unix est super!
~
~
~
~
~
~
"passwd" 17 lignes, 635 caractères
```

Comparez maintenant les deux fichiers :

```
# diff /etc/passwd /passwd
8d7
< nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/uucico
14d12
< svctag:x:95:12:Service Tag UID:/:
18c16,17
< test:x:100:1:Test:/export/home/test:/bin/ksh
---
> test10:x:100:1:Test:/export/home/test:/bin/ksh
> Unix est super!
```

Dans cette sortie on constate le caractère **<** et le caractère **>**. Le premier indique le premier fichier qui a suivi la commande **diff** tandis que le deuxième indique le deuxième fichier.

Le message **8d7** indique qu'il faut supprimer la ligne 8 dans /etc/passwd car elle n'existe pas après la ligne 7 dans le fichier /passwd.

Le message **14d12** indique qu'il faut supprimer la ligne 14 dans /etc/passwd car elle n'existe pas après la ligne 12 dans le fichier /passwd.

Le message **18c16,17** indique qu'il faut changer la ligne 18 dans /etc/passwd afin que celle-ci corresponde aux lignes 16 et 17.

Inversez maintenant l'ordre de comparaison :

```
# diff /passwd /etc/passwd
7a8
> nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/uucico
12a14
> svctag:x:95:12:Service Tag UID:/:
16,17c18
< test10:x:100:1:Test:/export/home/test:/bin/ksh
< Unix est super!
---
> test:x:100:1:Test:/export/home/test:/bin/ksh
```

<note important> Notez que les messages sont maintenant inversés. </note>

La Commande cmp

La commande **cmp** compare les fichiers caractère par caractère. Par défaut la commande s'arrête à la première différence rencontrée :

```
# cmp /passwd /etc/passwd
/passwd /etc/passwd diffèrent : car. 193, ligne 8
```

L'option **-l** de la commande indique toutes les différences en trois colonnes :

```
# cmp -l /passwd /etc/passwd | more
cmp : EOF sur /passwd
```

```
   193 163 156
   194 155 165
   195 155 165
   196 163 143
   201  62  71
   202  65  72
   203  72  71
   204  62  72
   205  65 165
   206  72 165
   207 123 143
   208 145 160
   209 156  40
   210 144 101
   211 115 144
   212 141 155
   214 154 156
   215  40  72
   216 115  57
   217 145 166
   218 163 141
   219 163 162
--A suivre--
```

La première colonne représente le numéro de caractère, la deuxième la valeur octale ASCII du caractère dans le fichier /passwd et la troisième la valeur octale ASCII du caractère dans le fichier /etc/passwd.

La commande patch

La commande **patch** est utilisée pour appliquer des modifications à un fichier à partir d'un fichier patch qui contient les différences entre le contenu de l'ancienne version du fichier et la nouvelle version.

Rappelez-vous des modifications apportées au fichier /tmp/greptest par rapport au fichier /tmp/greptest1 :

```
# cat /tmp/greptest
# Commentaire du début
^ voici une ligne pour la recherche fgrep
fenestrOS
fenestros
# Un autre commentaire
555-5555
f

.fenestros

.fe

£
# Commentaire de la fin
# cat /tmp/greptest1
fenestrOS
fenestros
555-5555
f
.fenestros
.fe
£
```

Afin de créer un fichier de patch, il convient d'utiliser la commande **diff** avec l'option **-u**

```
# cd /tmp
# diff -u greptest greptest1 > greptest.patch
```

L'examen du fichier de patch démontre les modifications à apporter au fichier **greptest** :

```
# cat greptest.patch
```

```
--- greptest    jeu. août 16 09:57:03 2012
+++ greptest1   jeu. août 16 09:54:15 2012
@@ -1,14 +1,7 @@
-# Commentaire du début
-^ voici une ligne pour la recherche fgrep
 fenestrOS
 fenestros
-# Un autre commentaire
 555-5555
 f
-
 .fenestros
-
 .fe
-
 £
-# Commentaire de la fin
```

Procédez maintenant à l'application du fichier patch :

```
# patch < greptest.patch
  Ressemble à une opération context diff unifiée.
Fichier auquel appliquer le patch : greptest
terminé
```

Contrôlez maintenant le contenu du fichier **greptest** :

```
# cat greptest
fenestrOS
fenestros
555-5555
f
.fenestros
.fe
```

```
£
```

La commande strings

La commande **strings** est utilisée pour trouver toutes les chaînes de caractères qui peuvent être imprimés dans un ou plusieurs fichiers objets ou exécutables passés en argument. Un fichier objet est un fichier intermédiaire intervenant dans le processus de compilation.

Sous Linux et Unix, le format d'un fichier objet est le format **ELF**, (*Executable and Linkable Format*). Ce format est aussi utilisé pour :

- les exécutables,
- les bibliothèques partagés,
- les core dumps.

Sans option, la commande **strings** trouve toutes les chaînes d'une longueur de 4 caractères ou plus suivies par un caractère non-imprimable :

```
# strings /usr/bin/passwd | more
usage:
nisplus
nisplus
nisplus
nisplus
nisplus
nisplus
nisplus
nisplus
nisplus
nisplus
nisplus
nisplus
/bin/sh
%s: %s
passwd -r ldap [-l|-N|-u] [-f] [-n min] [-w warn] [-x max] name
passwd -r ldap -s [name]
```

```
passwd -r ldap -sa
passwd -r ldap [-egh] [name]
[-x max] name
passwd -r nisplus [-D domainname] [-l|-N|-u] [-f] [-n min] [-w warn]
passwd -r nisplus [-D domainname] -s [name]
--A suivre--
```

L'option **-t** de la commande retourne, en plus des chaînes concernées, la position de décalage pour chaque ligne sur laquelle une ou plusieurs chaînes se trouvent :

```
# strings -t d /usr/bin/passwd | more
  16844 usage:
  16852 nisplus
  16860 nisplus
  16868 nisplus
  16876 nisplus
  16884 nisplus
  16892 nisplus
  16900 nisplus
  16908 nisplus
  16916 nisplus
  16924 nisplus
  16932 nisplus
  16940 nisplus
  16948 /bin/sh
  16956 %s: %s
  16965 passwd -r ldap [-l|-N|-u] [-f] [-n min] [-w warn] [-x max] name
  17033 passwd -r ldap -s [name]
  17061 passwd -r ldap -sa
  17085 passwd -r ldap [-egh] [name]
  17118 [-x max] name
  17137 passwd -r nisplus [-D domainname] [-l|-N|-u] [-f] [-n min] [-w warn]
  17209 passwd -r nisplus [-D domainname] -s [name]
```

```
--A suivre--
```

L'option **-t** prend un de trois arguments qui indique le système de numérotation à utiliser :

Argument	Système de Numérotation
d	Décimal
o	Octal
x	Héxadécimal

L'option **-n** de la commande permet de modifier le nombre de caractères minimales dans les chaînes recherchées :

```
# strings -t d -n 15 /usr/bin/passwd | more
  16965 passwd -r ldap [-l|-N|-u] [-f] [-n min] [-w warn] [-x max] name
  17033 passwd -r ldap -s [name]
  17061 passwd -r ldap -sa
  17085 passwd -r ldap [-egh] [name]
  17137 passwd -r nisplus [-D domainname] [-l|-N|-u] [-f] [-n min] [-w warn]
  17209 passwd -r nisplus [-D domainname] -s [name]
  17257 passwd -r nisplus -sa
  17281 passwd -r nisplus [-egh] [-D domainname] [name]
  17333 passwd -r nis [-eg] [name]
  17365 passwd [-r files] [-d|-l|-N|-u] [-f] [-n min] [-w warn] [-x max] name
  17437 passwd [-r files] -s [name]
  17469 passwd [-r files] -sa
  17493 passwd [-r files] [-egh] [name]
  17529 passwd [-r files | -r nis | -r nisplus | -r ldap] [name]
  17612 Permission denied
  17648 Password aging is disabled
  17676 Invalid argument to option
  17704 Password file/table busy. Try again later.
  17748 Unexpected failure. Password file/table missing.
  17800 Unexpected failure. Password file/table unchanged.
  17852 Invalid combination of options
  17884 Permission denied
```

```
--A suivre--
```

La commande comm

La commande **comm** est utilisée pour comparer deux fichiers texte. La sortie de la commande sépare les lignes en trois catégories :

- Les lignes présentes seulement dans le premier fichier,
- Les lignes présentes seulement dans le deuxième fichier,
- Les lignes présentes dans la deux fichiers.

Utilisez la commande **comm** pour comparer les fichiers **/etc/passwd** et **/root/passwd** :

```
# comm /etc/passwd /passwd
                root:x:0:0:Super-User:/:/sbin/sh
                daemon:x:1:1::/:
                bin:x:2:2::/usr/bin:
                sys:x:3:3::/:
                adm:x:4:4:Admin:/var/adm:
                lp:x:71:8:Line Printer Admin:/usr/spool/lp:
                uucp:x:5:5:uucp Admin:/usr/lib/uucp:
nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/uucico
                smmsp:x:25:25:SendMail Message Submission Program:/:
                listen:x:37:4:Network Admin:/usr/net/nls:
                gdm:x:50:50:GDM Reserved UID:/:
                webservd:x:80:80:WebServer Reserved UID:/:
                postgres:x:90:90:PostgreSQL Reserved UID:/:/usr/bin/pfksh
        nobody:x:60001:60001:NFS Anonymous Access User:/:
        noaccess:x:60002:60002:No Access User:/:
        nobody4:x:65534:65534:SunOS 4.x NFS Anonymous Access User:/:
svctag:x:95:12:Service Tag UID:/:
nobody:x:60001:60001:NFS Anonymous Access User:/:
noaccess:x:60002:60002:No Access User:/:
```

```
nobody4:x:65534:65534:SunOS 4.x NFS Anonymous Access User:/:
test:x:100:1:Test:/export/home/test:/bin/ksh
        test10:x:100:1:Test:/export/home/test:/bin/ksh
        Unix est super!
```

Pour afficher uniquement les lignes présentes dans les deux fichiers, il convient d'utiliser les options **-1** et **-2** :

```
# comm -12 /etc/passwd /passwd
root:x:0:0:Super-User:/:/sbin/sh
daemon:x:1:1::/:
bin:x:2:2::/usr/bin:
sys:x:3:3::/:
adm:x:4:4:Admin:/var/adm:
lp:x:71:8:Line Printer Admin:/usr/spool/lp:
uucp:x:5:5:uucp Admin:/usr/lib/uucp:
smmsp:x:25:25:SendMail Message Submission Program:/:
listen:x:37:4:Network Admin:/usr/net/nls:
gdm:x:50:50:GDM Reserved UID:/:
webservd:x:80:80:WebServer Reserved UID:/:
postgres:x:90:90:PostgreSQL Reserved UID:/:/usr/bin/pfksh
```

La commande head

La commande **head** permet d'afficher les **x** premières lignes d'un fichier. Sans options, la valeur de **x** est de 10 par défaut :

```
# head /etc/passwd
root:x:0:0:Super-User:/:/sbin/sh
daemon:x:1:1::/:
bin:x:2:2::/usr/bin:
sys:x:3:3::/:
adm:x:4:4:Admin:/var/adm:
lp:x:71:8:Line Printer Admin:/usr/spool/lp:
```

```
uucp:x:5:5:uucp Admin:/usr/lib/uucp:
nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/uucico
smmsp:x:25:25:SendMail Message Submission Program:/:
listen:x:37:4:Network Admin:/usr/net/nls:
```

Avec l'option **-n**, la valeur de **x** peut être spécifiée :

```
# head -n 12 /etc/passwd
root:x:0:0:Super-User:/:/sbin/sh
daemon:x:1:1::/:
bin:x:2:2::/usr/bin:
sys:x:3:3::/:
adm:x:4:4:Admin:/var/adm:
lp:x:71:8:Line Printer Admin:/usr/spool/lp:
uucp:x:5:5:uucp Admin:/usr/lib/uucp:
nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/uucico
smmsp:x:25:25:SendMail Message Submission Program:/:
listen:x:37:4:Network Admin:/usr/net/nls:
gdm:x:50:50:GDM Reserved UID:/:
webservd:x:80:80:WebServer Reserved UID:/:
```

La commande tail

La commande **tail** permet d'afficher les **x** dernières lignes d'un fichier. Sans options, la valeur de **x** est de 10 par défaut :

```
# tail /etc/passwd
smmsp:x:25:25:SendMail Message Submission Program:/:
listen:x:37:4:Network Admin:/usr/net/nls:
gdm:x:50:50:GDM Reserved UID:/:
webservd:x:80:80:WebServer Reserved UID:/:
postgres:x:90:90:PostgreSQL Reserved UID:/:/usr/bin/pfksh
svctag:x:95:12:Service Tag UID:/:
```

```
nobody:x:60001:60001:NFS Anonymous Access User:/:
noaccess:x:60002:60002:No Access User:/:
nobody4:x:65534:65534:SunOS 4.x NFS Anonymous Access User:/:
test:x:100:1:Test:/export/home/test:/bin/ksh
```

La valeur de **x** peut être spécifiée :

```
# tail -15 /passwd
bin:x:2:2::/usr/bin:
sys:x:3:3::/:
adm:x:4:4:Admin:/var/adm:
lp:x:71:8:Line Printer Admin:/usr/spool/lp:
uucp:x:5:5:uucp Admin:/usr/lib/uucp:
smmsp:x:25:25:SendMail Message Submission Program:/:
listen:x:37:4:Network Admin:/usr/net/nls:
gdm:x:50:50:GDM Reserved UID:/:
webservd:x:80:80:WebServer Reserved UID:/:
postgres:x:90:90:PostgreSQL Reserved UID:/:/usr/bin/pfksh
nobody:x:60001:60001:NFS Anonymous Access User:/:
noaccess:x:60002:60002:No Access User:/:
nobody4:x:65534:65534:SunOS 4.x NFS Anonymous Access User:/:
test10:x:100:1:Test:/export/home/test:/bin/ksh
Unix est super!
```

Une option intéressante pour la surveillance des fichiers de journalisation est **-f**. Cette option met à jour l'affichage au fur et au mesure que le fichier est mis à jour :

```
# tail -f /var/adm/messages
Aug 14 16:19:31 unknown pseudo: [ID 129642 kern.info] pseudo-device: winlock0
Aug 14 16:19:31 unknown genunix: [ID 936769 kern.info] winlock0 is /pseudo/winlock@0
Aug 14 16:19:31 unknown pseudo: [ID 129642 kern.info] pseudo-device: pm0
Aug 14 16:19:31 unknown genunix: [ID 936769 kern.info] pm0 is /pseudo/pm@0
Aug 14 16:19:31 unknown pseudo: [ID 129642 kern.info] pseudo-device: rsm0
Aug 14 16:19:31 unknown genunix: [ID 936769 kern.info] rsm0 is /pseudo/rsm@0
```

```
Aug 14 17:24:20 unknown e1000g: [ID 801725 kern.info] NOTICE: pci8086,100e - e1000g[0] : link down
Aug 14 17:24:25 unknown e1000g: [ID 801725 kern.info] NOTICE: pci8086,100e - e1000g[0] : link up, 1000 Mbps, full
duplex
Aug 14 17:24:25 unknown /sbin/dhcpagent[101]: [ID 967406 daemon.warning] refreshing state on e1000g0
Aug 14 17:24:25 unknown /sbin/dhcpagent[101]: [ID 778557 daemon.warning] configure_v4_lease: no IP broadcast
specified for e1000g0, making best guess
^C#
```

<html> Copyright © 2019 Hugh Norris. </html>