

Chapitre 4 : pipes, redirections et expressions régulières

1

Introduction

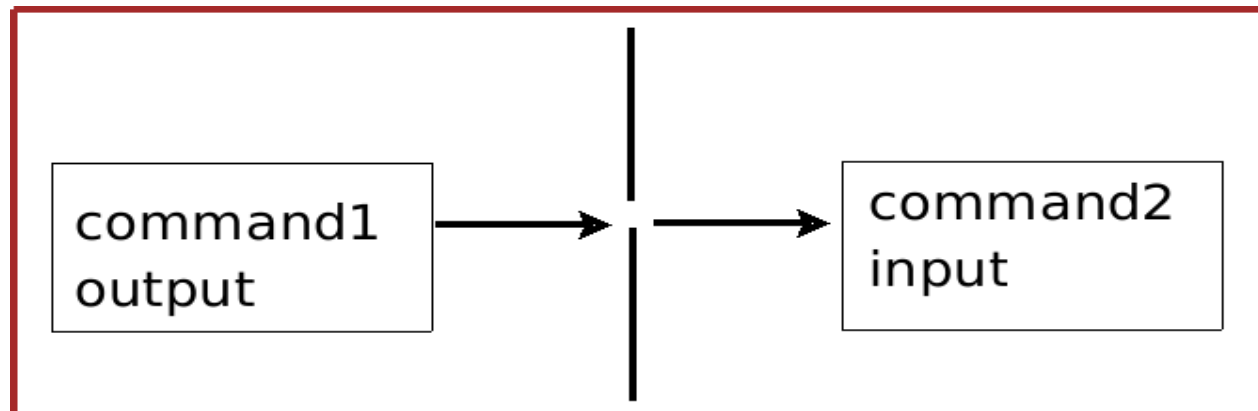
- La plupart des fichiers à manipuler sur un système Linux sont des fichiers texte (exemple : fichier de configuration) qui contiennent un simple texte.
- De ce fait le Shell Linux offre plusieurs outils permettant de manipuler les fichiers textes (affichage , édition, recherche...)
- En plus le Shell possède des moyens permettant de contrôler la sortie d'une commande au lieu d'afficher cette sortie sur la fenêtre du terminal on peut la rediriger vers un fichier ou vers une autre commande.

Objectifs

- Utilisation des pipes
- Utilisation des redirections des commandes
- Manipuler les fichiers textes
- Utilisation des expressions régulières

Les pipes

- Dans le chapitre précédent on a vu comment utiliser des commandes individuels pour faire une tâche(lister , copier , déplacer...), généralement lorsqu'une commande génère un résultat ou une erreur le résultat (ou l'erreur) est affiché à l'écran
- Le caractère pipe (|) est utilisé pour envoyer la sortie (résultat) d'une commande à une autre au lieu de l'afficher à l'écran , ainsi la sortie d'une commande devient une entrée pour une autre commande

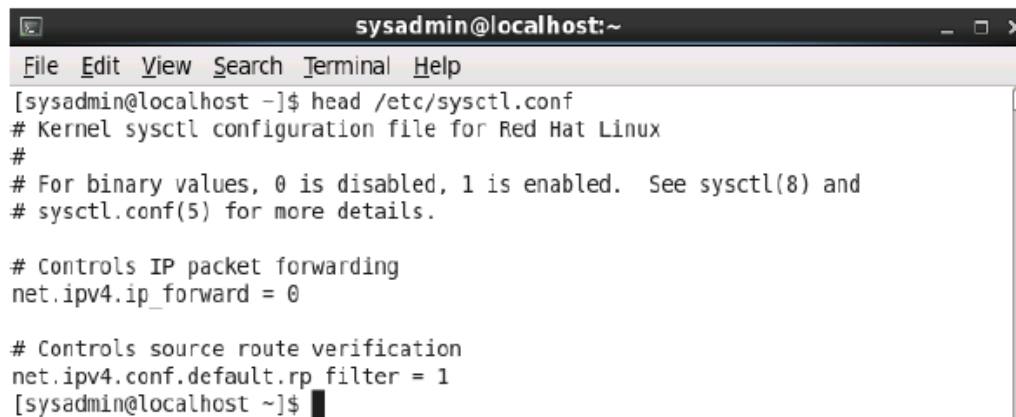


Les pipes

- Le pipe est un outil puissant il est surtout utiliser pour raffiner et filtrer le résultat d'une commande initial
- Pour illustrer le mécanisme du piping dans les exemples qui vont suivre on va utiliser les commandes **head** et **tail**
 - ***head***: permet d'afficher les premières ligne d'un fichier
 - ***tail***: permet d'afficher les dernières lignes d'un fichier
 - Par défaut les commandes ***head*** et ***tail*** affichent 10 lignes

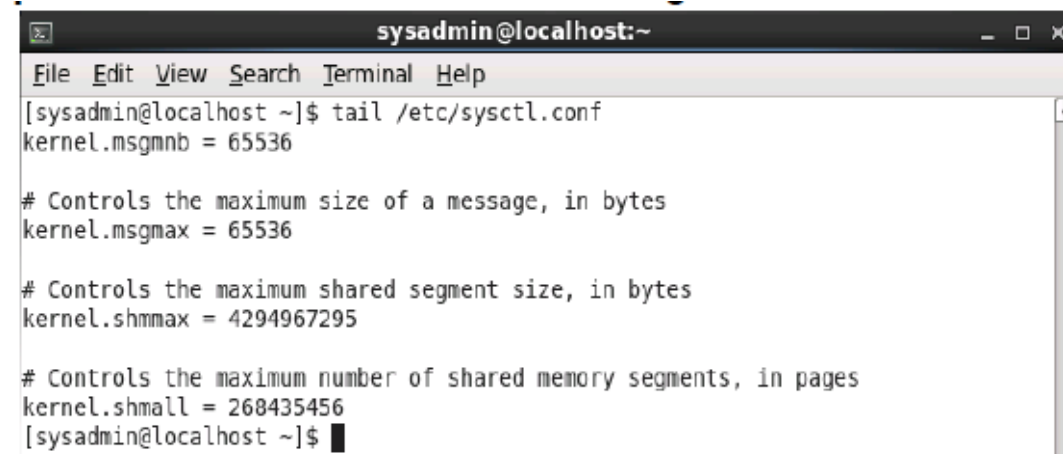
Les pipes

- Exemple1 : dans cet exemple on affiche les 10 premières lignes du fichier /etc/sysctl.conf



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ head /etc/sysctl.conf  
# Kernel sysctl configuration file for Red Hat Linux  
#  
# For binary values, 0 is disabled, 1 is enabled. See sysctl(8) and  
# sysctl.conf(5) for more details.  
  
# Controls IP packet forwarding  
net.ipv4.ip_forward = 0  
  
# Controls source route verification  
net.ipv4.conf.default.rp_filter = 1  
[sysadmin@localhost ~]$
```

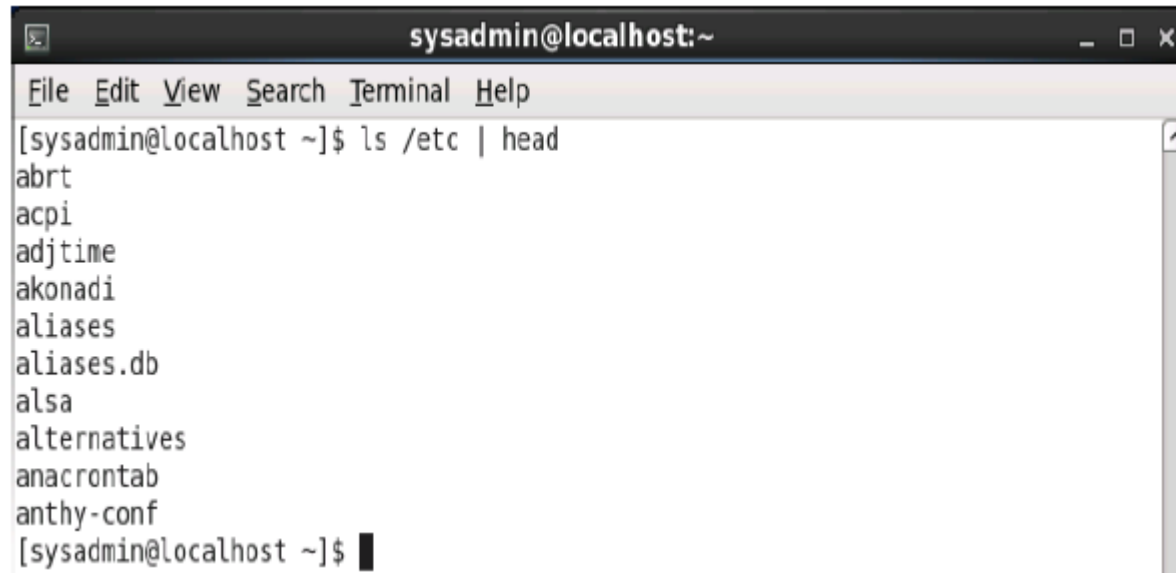
- Exemple2 : ici on affiche les 10 dernières lignes du même fichier



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ tail /etc/sysctl.conf  
kernel.msgmnb = 65536  
  
# Controls the maximum size of a message, in bytes  
kernel.msgmax = 65536  
  
# Controls the maximum shared segment size, in bytes  
kernel.shmmax = 4294967295  
  
# Controls the maximum number of shared memory segments, in pages  
kernel.shmall = 268435456  
[sysadmin@localhost ~]$
```

Les pipes

- Les commandes head et tail ne s'appliquent pas uniquement aux fichiers mais on peut les appliquer aussi sur la sortie d'une commande
- – Exemple 3:ici on a passé à la commande head la sortie de la commande ls /etc,comme résultat on va lister uniquement les 10 premiers fichiers contenus dans /etc et non la totalité des fichiers

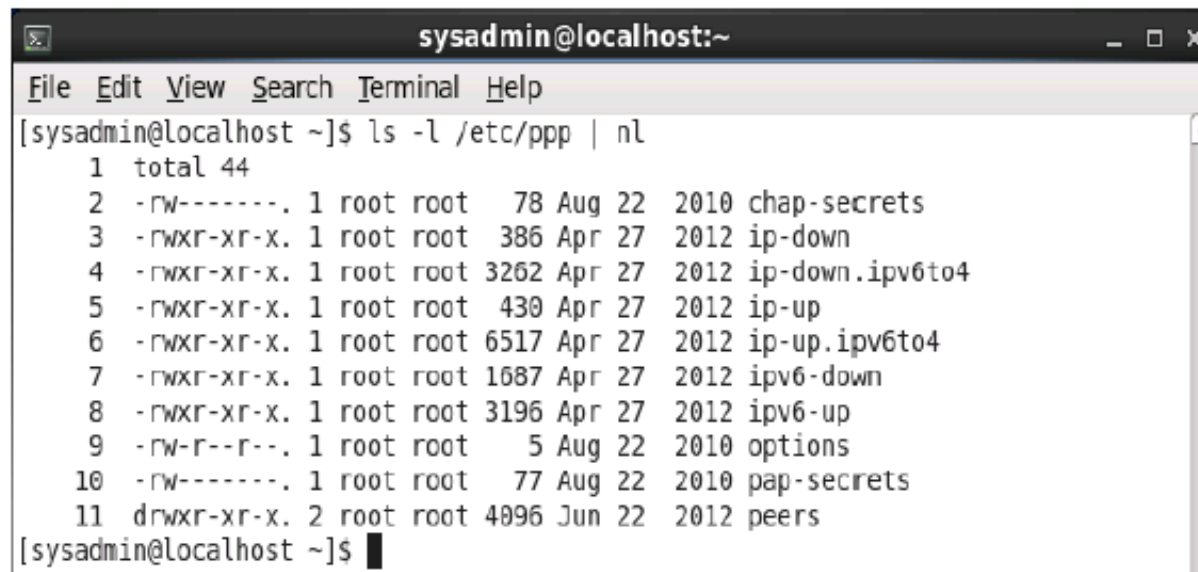


```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ ls /etc | head  
abrt  
acpi  
adjtime  
akonadi  
aliases  
aliases.db  
alsa  
alternatives  
anacrontab  
anthy-conf  
[sysadmin@localhost ~]$
```

- – Le résultat de ls /etc n'est pas affiché à l'écran mais sert d'entrée à la commande head qui va faire son traitement puis afficher le résultat à l'écran

Les pipes

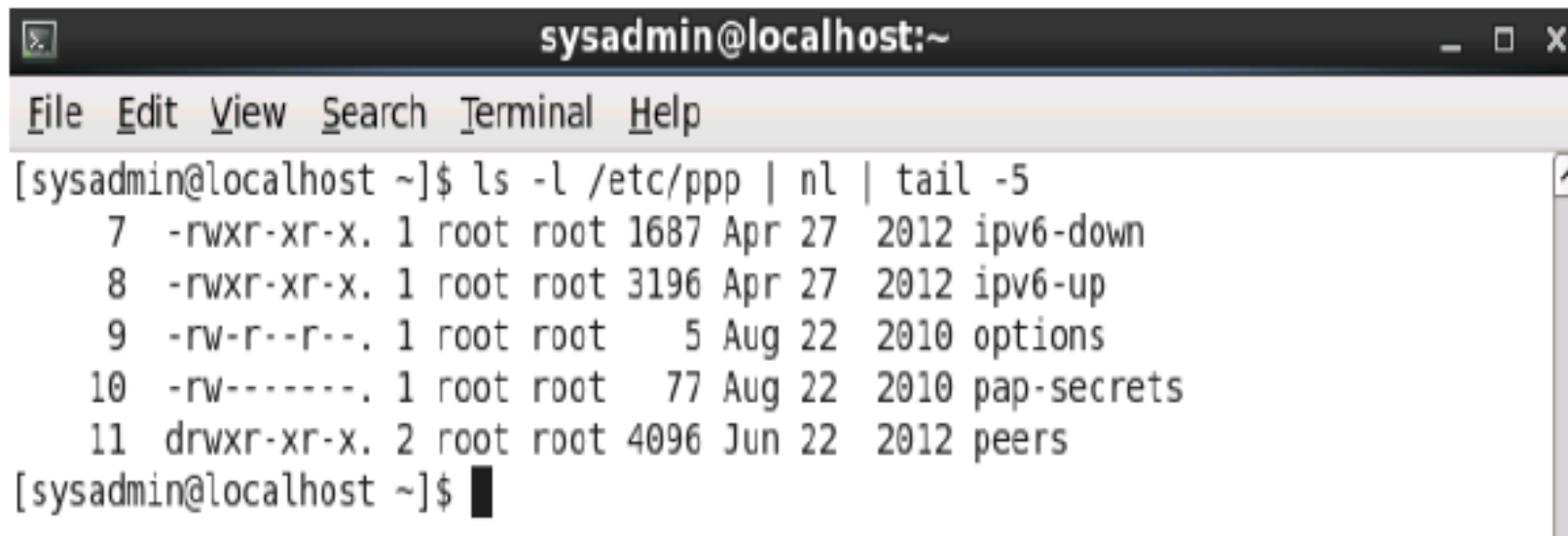
- Il est important de bien choisir l'ordre des commandes lorsqu'on utilise le piping pour illustrer l'importance de l'ordre voyons les exemples suivants:
- Exemple 4 : ici on a passé la sortie de la commande `ls -l /etc/ppp` à la commande `nl` (`nl` permet de numéroter les lignes de la sortie d'une commande)



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ ls -l /etc/ppp | nl  
1 total 44  
2 -rw-----. 1 root root 78 Aug 22 2010 chap-secrets  
3 -rwxr-xr-x. 1 root root 386 Apr 27 2012 ip-down  
4 -rwxr-xr-x. 1 root root 3262 Apr 27 2012 ip-down.ipv6to4  
5 -rwxr-xr-x. 1 root root 430 Apr 27 2012 ip-up  
6 -rwxr-xr-x. 1 root root 6517 Apr 27 2012 ip-up.ipv6to4  
7 -rwxr-xr-x. 1 root root 1687 Apr 27 2012 ipv6-down  
8 -rwxr-xr-x. 1 root root 3196 Apr 27 2012 ipv6-up  
9 -rw-r--r--. 1 root root 5 Aug 22 2010 options  
10 -rw-----. 1 root root 77 Aug 22 2010 pap-secrets  
11 drwxr-xr-x. 2 root root 4096 Jun 22 2012 peers  
[sysadmin@localhost ~]$
```

Les pipes

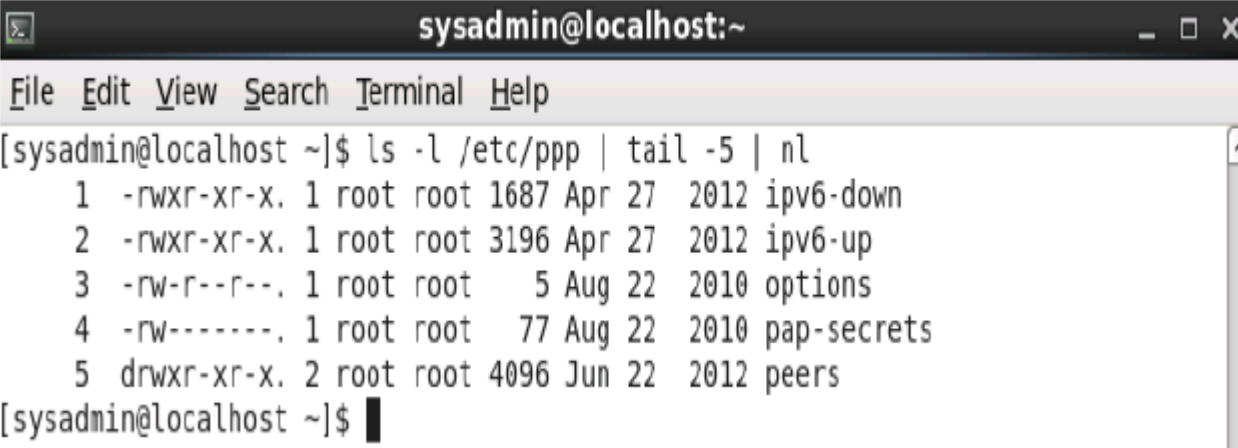
- Exemple 5 : dans cet exemple la commande `ls -l /etc/ppp` est exécutée puis son résultat est passé à la commande `nl` qui va numéroter les lignes du résultat de la commande précédente puis on a passé le résultat de `nl` à `tail -5` qui va afficher sur l'écran les 5 dernières lignes du résultat de `nl`



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ ls -l /etc/ppp | nl | tail -5  
7 -rwxr-xr-x. 1 root root 1687 Apr 27 2012 ipv6-down  
8 -rwxr-xr-x. 1 root root 3196 Apr 27 2012 ipv6-up  
9 -rw-r--r--. 1 root root 5 Aug 22 2010 options  
10 -rw-----. 1 root root 77 Aug 22 2010 pap-secrets  
11 drwxr-xr-x. 2 root root 4096 Jun 22 2012 peers  
[sysadmin@localhost ~]$
```

Les pipes

- Exemple 6 : le même exemple précédent mais on a inversé l'ordre des commandes **nl** et **tail -5**



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ ls -l /etc/ppp | tail -5 | nl  
1 -rwxr-xr-x. 1 root root 1687 Apr 27 2012 ipv6-down  
2 -rwxr-xr-x. 1 root root 3196 Apr 27 2012 ipv6-up  
3 -rw-r--r--. 1 root root 5 Aug 22 2010 options  
4 -rw-----. 1 root root 77 Aug 22 2010 pap-secrets  
5 drwxr-xr-x. 2 root root 4096 Jun 22 2012 peers  
[sysadmin@localhost ~]$
```

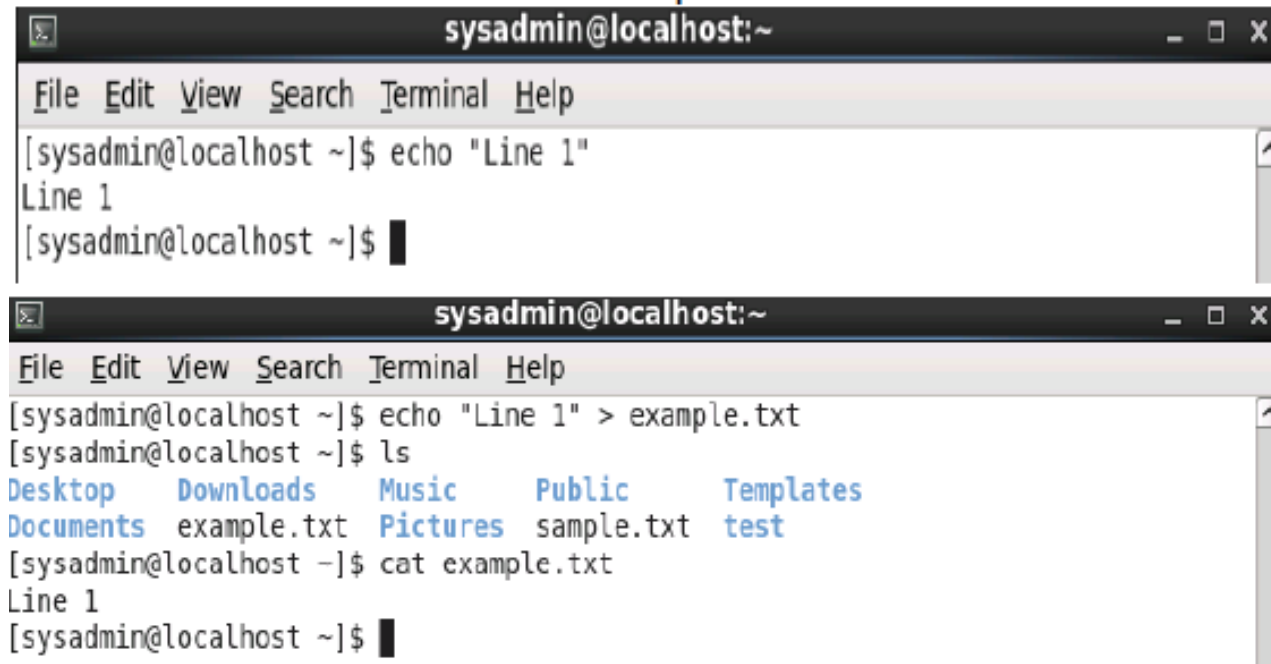
- Ici le résultat de ls est d'abord envoyé à tail -5 qui va prendre 5 lignes de ce résultat puis il va les passer à nl qui va les numéroter de 1 à 5.



- La redirection des entrées / sorties permet de rediriger la sortie d'une commande vers différents flux de données (écran, fichier...) et aussi de rediriger un flux de données vers l'entrée d'une commande (clavier, fichier...)
- Normalement une commande reçoit les données à partir de l'entrée standard (STDIN) qui est le clavier et affiche son résultat sur la sortie standard (STDOUT) qui est l'écran
- Il existe un troisième flux qui est la sortie standard d'erreur (STDERR), une commande lorsqu'elle génère une erreur celle-ci est affichée sur l'écran (STDERR)
- On peut rediriger l'entrée standard d'une commande, la commande va lire les données à partir d'un fichier et non à partir du clavier
- Comme on peut aussi rediriger la sortie d'une commande (STDOUT/STDERR) vers un fichier, la commande va écrire son résultat ou erreur sur un fichier et ne va pas les afficher sur l'écran

Redirection de la sortie standard (STDOUT)

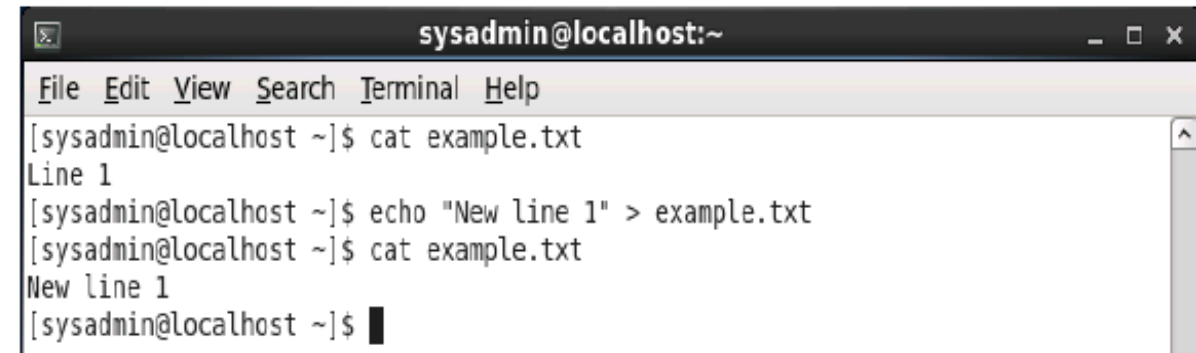
- Pour rediriger la sortie d'une commande vers un fichier on utilise le symbole >
 - **commande > nom-fichier**
 - Exemple 1: on a rediriger la sortie standard(STDOUT) de la commande echo vers le fichier example.txt donc le résultat ne sera pas afficher sur l'écran mais écrit sur le fichier example.txt



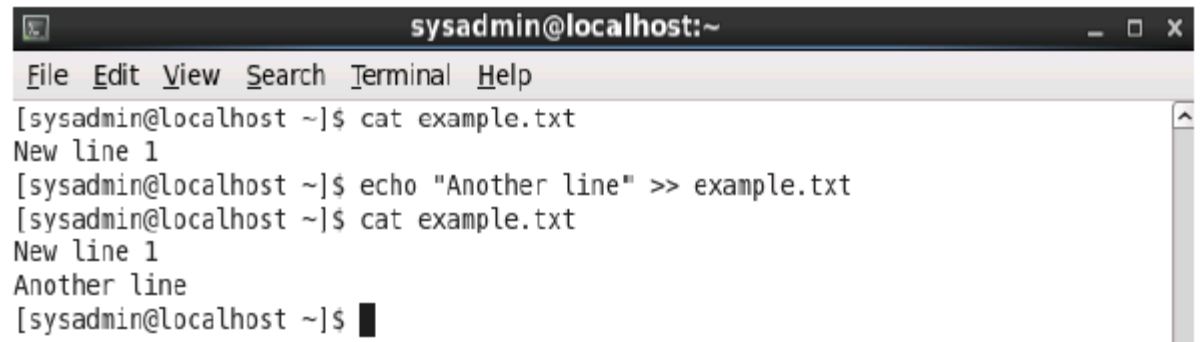
```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ echo "Line 1"  
Line 1  
[sysadmin@localhost ~]$  
  
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ echo "Line 1" > example.txt  
[sysadmin@localhost ~]$ ls  
Desktop Downloads Music Public Templates  
Documents example.txt Pictures sample.txt test  
[sysadmin@localhost ~]$ cat example.txt  
Line 1  
[sysadmin@localhost ~]$
```

Redirection de la sortie standard (STDOUT)

- Lorsqu'on utilise un seul symbole ">" le contenu du fichier sera écrasé, pour éviter cela il faut utiliser un double symbole ">>"
- Exemple 2:ici le contenu du fichier example.txt sera écrasé lorsqu'on redirige la sortie standard de echo vers ce dernier en utilisant un seul caractère ">"
- Ici le contenu du fichier sera conservé et la sortie de la commande echo sera ajoutée à la fin du fichier grâce à l'utilisation du double supérieur ">>"



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ cat example.txt  
Line 1  
[sysadmin@localhost ~]$ echo "New line 1" > example.txt  
[sysadmin@localhost ~]$ cat example.txt  
New line 1  
[sysadmin@localhost ~]$
```



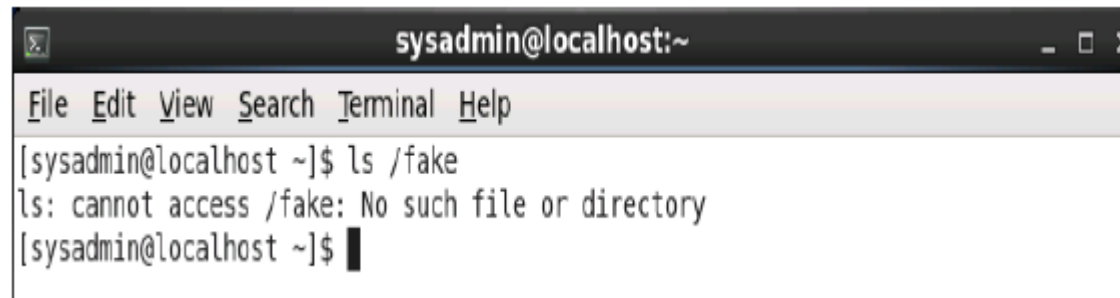
```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ cat example.txt  
New line 1  
[sysadmin@localhost ~]$ echo "Another line" >> example.txt  
[sysadmin@localhost ~]$ cat example.txt  
New line 1  
Another line  
[sysadmin@localhost ~]$
```

Redirection de la sortie d'erreur standard (STDERR)

- La sortie d'erreur standard(STDERR) d'une commande peut être redirigé vers un fichier de la même manière qu'une sortie standard(STDOUT)
- Le flux de données de la sortie standard est identifié par le numéro 1,alors que le flux de la sortie d'erreur est identifié par 2.
- Lorsqu'on utilise le caractère “>” pour rediriger un flux(STDOUT,STDERR) sans indiquer de numéro de flux(1,2) c'est STDOUT qui est redirigée par défaut
- Pour rediriger STDERR il faut alors indiquer le numéro de flux 2,examinons les exemples suivants :

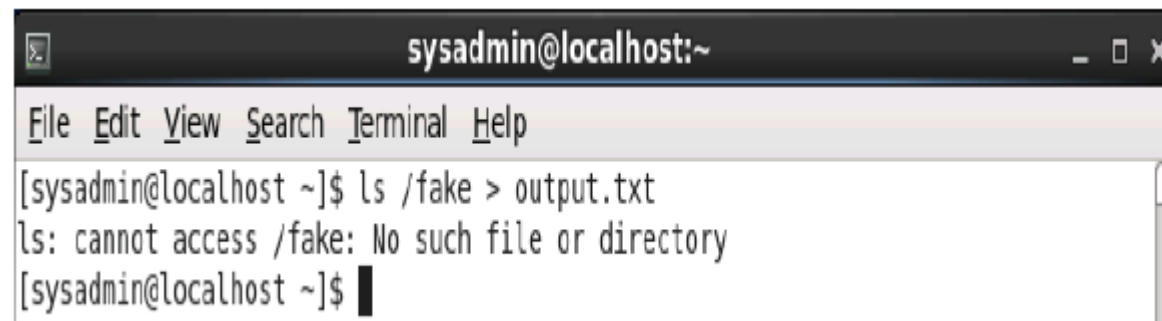
Redirection de la sortie d'erreur standard (STDERR)

- Exemple 1:
 - on commence par exécuter une commande qui va générer une erreur

A terminal window titled 'sysadmin@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is '[sysadmin@localhost ~]\$'. The command 'ls /fake' has been entered, and the output is 'ls: cannot access /fake: No such file or directory'. The prompt '[sysadmin@localhost ~]\$' is shown again with a cursor.

```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ ls /fake  
ls: cannot access /fake: No such file or directory  
[sysadmin@localhost ~]$
```

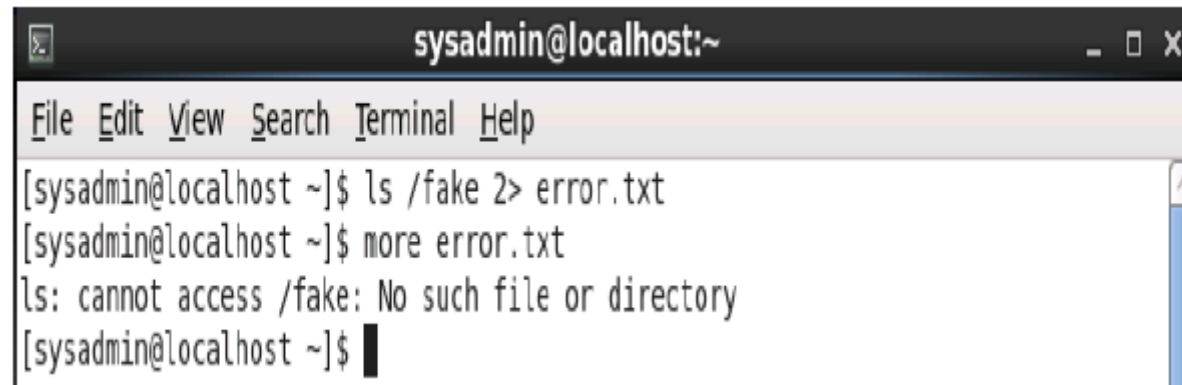
- Ensuite on va rediriger la sortie de cette commande en utilisant le caractère ">", on remarque que le message d'erreur est affiché sur l'écran car c'est STDOUT qui est redirigé par défaut et non STDERR

A terminal window titled 'sysadmin@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is '[sysadmin@localhost ~]\$'. The command 'ls /fake > output.txt' has been entered. The output is 'ls: cannot access /fake: No such file or directory'. The prompt '[sysadmin@localhost ~]\$' is shown again with a cursor.

```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ ls /fake > output.txt  
ls: cannot access /fake: No such file or directory  
[sysadmin@localhost ~]$
```


Redirection de la sortie d'erreur standard (STDERR)

- Exemple 1:(suite)
 - Pour rediriger STDERR il faut utilisé “2>” qui signifie rediriger le flux de données de la sortie d'erreur on écrit `ls /fake 2> error.txt`

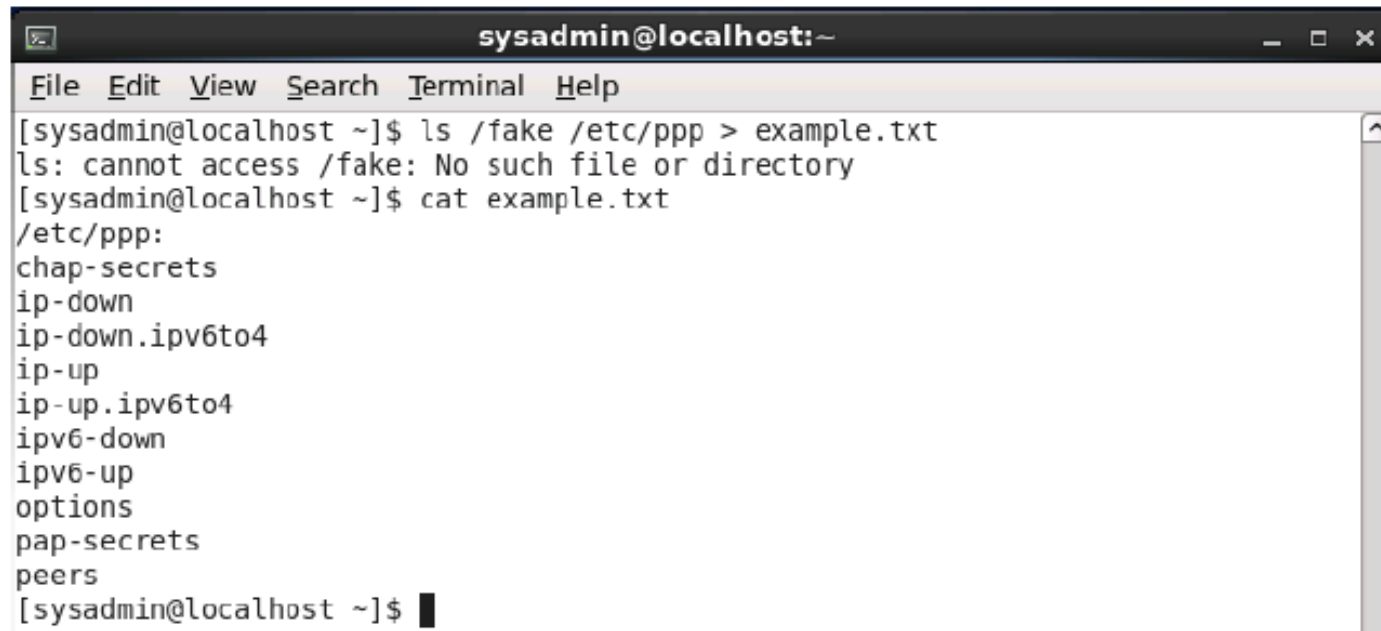
A terminal window titled 'sysadmin@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the following commands and output:

```
[sysadmin@localhost ~]$ ls /fake 2> error.txt
[sysadmin@localhost ~]$ more error.txt
ls: cannot access /fake: No such file or directory
[sysadmin@localhost ~]$
```

- Ici par exemple l'erreur affichée par la commande `ls /fake` sera écrite(redirigée) vers le fichier `error.txt`

Redirection de flux multiples

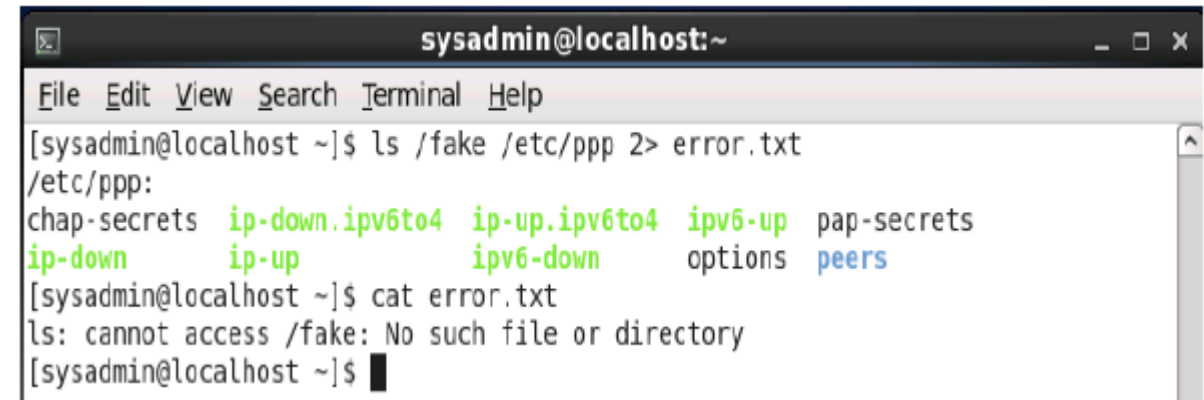
- Il est possible de rediriger STDOUT et STDERR d'une commande au même temps
- Exemple:
 - Lorsqu'on utilise le caractère ">" c'est uniquement STDOUT qui est redirigée et STDERR est affichée sur l'écran comme le montre l'exemple.

A terminal window titled 'sysadmin@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the following commands and output:

```
[sysadmin@localhost ~]$ ls /fake /etc/ppp > example.txt
ls: cannot access /fake: No such file or directory
[sysadmin@localhost ~]$ cat example.txt
/etc/ppp:
chap-secrets
ip-down
ip-down.ipv6to4
ip-up
ip-up.ipv6to4
ipv6-down
ipv6-up
options
pap-secrets
peers
[sysadmin@localhost ~]$
```

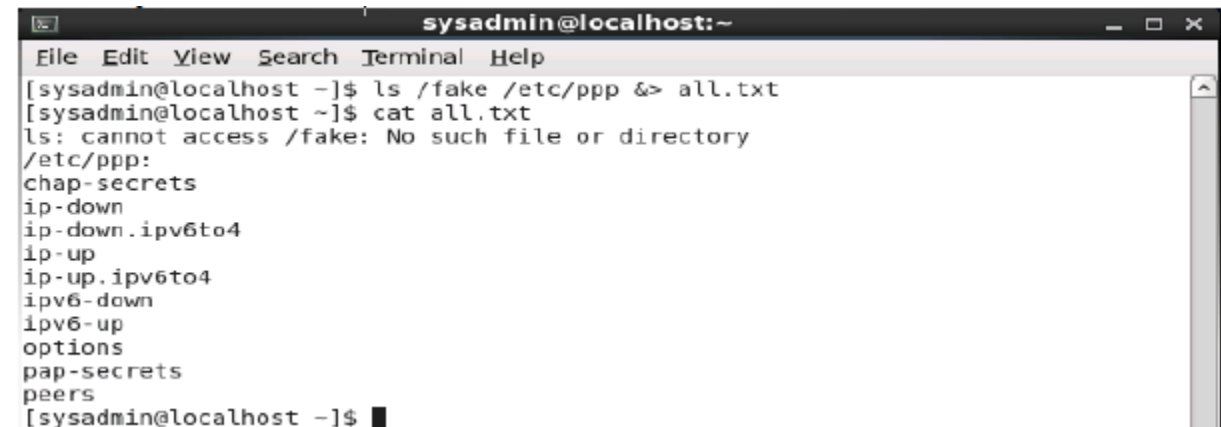
Redirection de flux multiples

- Exemple (suite):
 - Lorsqu'on utilise le symbole “2>” c'est uniquement la sortie d'erreur qui est redirigée et STDOUT est affichée sur l'écran



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ ls /fake /etc/ppp 2> error.txt  
/etc/ppp:  
chap-secrets  ip-down.ipv6to4  ip-up.ipv6to4  ipv6-up  pap-secrets  
ip-down      ip-up            ipv6-down      options  peers  
[sysadmin@localhost ~]$ cat error.txt  
ls: cannot access /fake: No such file or directory  
[sysadmin@localhost ~]$
```

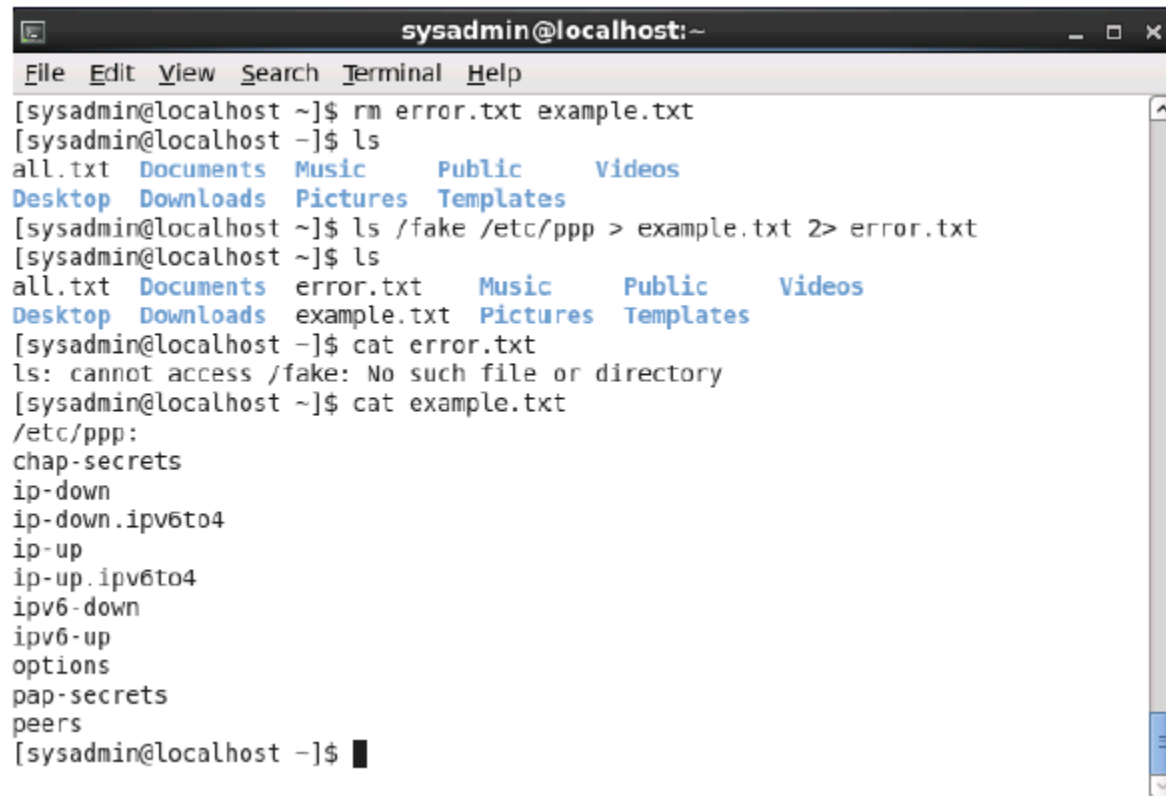
- Pour rediriger au même temps STDOUT et STDERR vers un fichier on utilise le symbole “&>”



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ ls /fake /etc/ppp &> all.txt  
[sysadmin@localhost ~]$ cat all.txt  
ls: cannot access /fake: No such file or directory  
/etc/ppp:  
chap-secrets  
ip-down  
ip-down.ipv6to4  
ip-up  
ip-up.ipv6to4  
ipv6-down  
ipv6-up  
options  
pap-secrets  
peers  
[sysadmin@localhost ~]$
```

Redirection de flux multiples

- Pour envoyer les deux flux (STDOUT et STDERR) vers deux fichiers différents on utilise les deux symboles “>” et “2>” comme le montre l'exemple suivant:

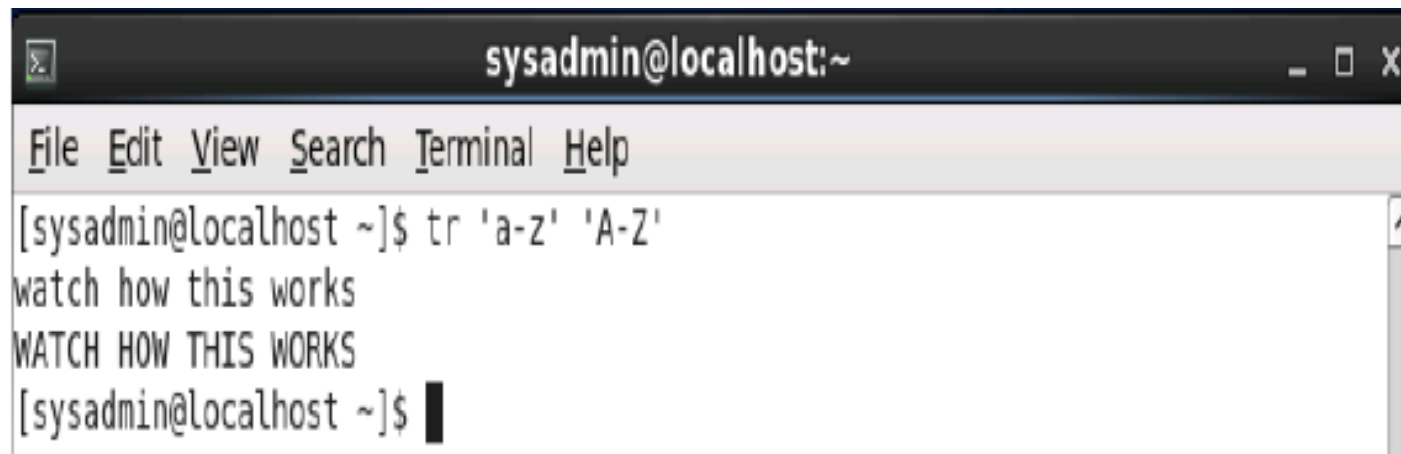
A terminal window titled 'sysadmin@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows a sequence of commands: 'rm error.txt example.txt', 'ls' (listing standard directories), 'ls /fake /etc/ppp > example.txt 2> error.txt', and another 'ls' (listing 'example.txt' and 'error.txt'). It then shows 'cat error.txt' (outputting an 'ls' error message) and 'cat example.txt' (outputting the contents of '/etc/ppp').

```
[sysadmin@localhost ~]$ rm error.txt example.txt
[sysadmin@localhost ~]$ ls
all.txt  Documents  Music      Public     Videos
Desktop  Downloads  Pictures   Templates
[sysadmin@localhost ~]$ ls /fake /etc/ppp > example.txt 2> error.txt
[sysadmin@localhost ~]$ ls
all.txt  Documents  error.txt  Music      Public     Videos
Desktop  Downloads  example.txt  Pictures   Templates
[sysadmin@localhost ~]$ cat error.txt
ls: cannot access /fake: No such file or directory
[sysadmin@localhost ~]$ cat example.txt
/etc/ppp:
chap-secrets
ip-down
ip-down.ipv6to4
ip-up
ip-up.ipv6to4
ipv6-down
ipv6-up
options
pap-secrets
peers
[sysadmin@localhost ~]$
```

- L'ordre dans lequel on spécifie les deux flux n'a pas d'influence

Redirection de l'entrée standard (STDIN)

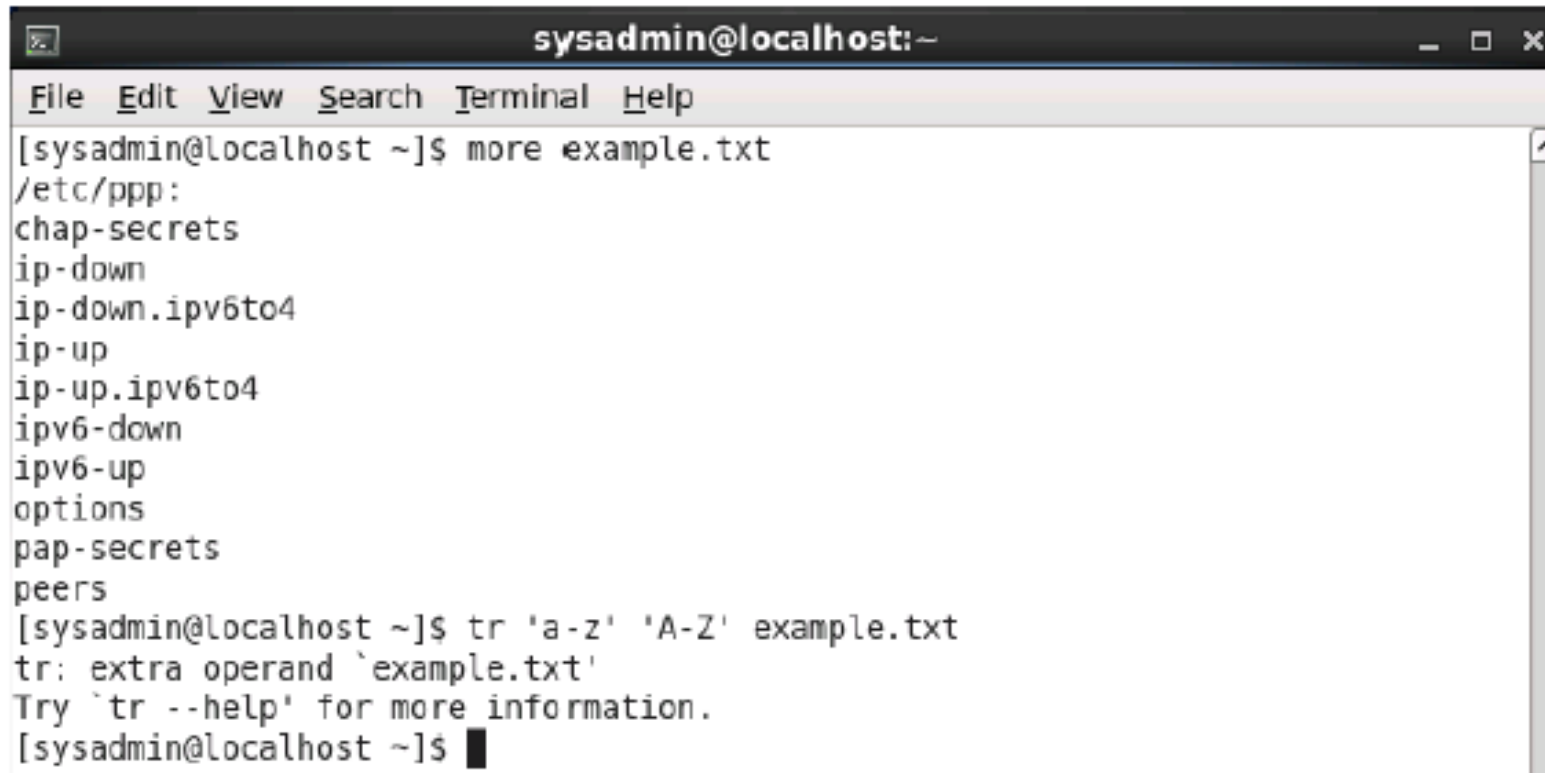
- La redirection de l'entrée standard (STDIN 0) consiste à faire passer à une commande les données à partir d'un fichier et non à partir du clavier
- Pour donner des exemples de redirection de STDIN on considère la commande `tr` qui permet de traduire une suite de caractères en une autre suite de caractères
 - Exemple : on souhaite mettre en majuscule un texte qui est écrit en minuscule on doit utiliser la commande `tr` comme suit



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ tr 'a-z' 'A-Z'  
watch how this works  
WATCH HOW THIS WORKS  
[sysadmin@localhost ~]$
```

Redirection de l'entrée standard (STDIN)

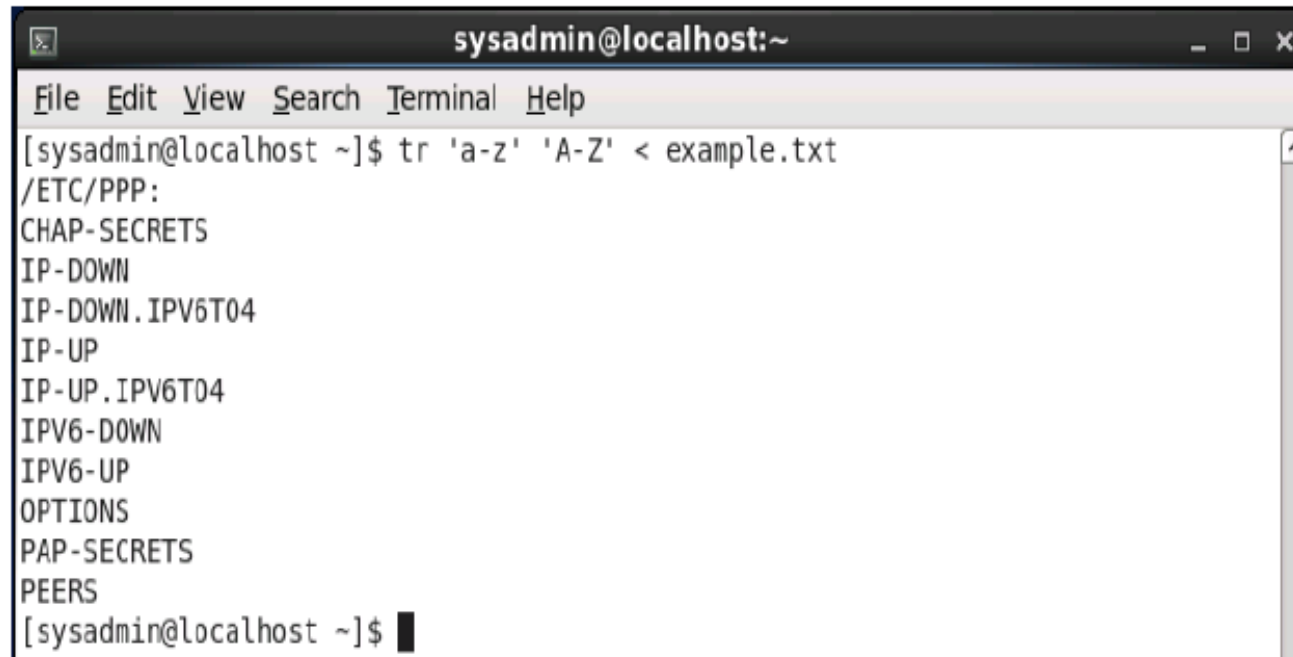
- L'exemple précédent montre que la commande `tr` lit les données à partir du clavier , pour forcer `tr` à lire les données à partir d'un fichier on utilise le caractère de redirection "<"
 - Exemple : ici on affiche le contenu du fichier `example.txt` qu'on souhaite passer à l'entrée de la commande `tr`

A terminal window titled 'sysadmin@localhost:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command 'more example.txt' being executed, displaying the contents of the file. Then, the command 'tr 'a-z' 'A-Z' example.txt' is entered, but it fails with an error message: 'tr: extra operand `example.txt' Try `tr --help' for more information.'

```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ more example.txt  
/etc/ppp:  
chap-secrets  
ip-down  
ip-down.ipv6to4  
ip-up  
ip-up.ipv6to4  
ipv6-down  
ipv6-up  
options  
pap-secrets  
peers  
[sysadmin@localhost ~]$ tr 'a-z' 'A-Z' example.txt  
tr: extra operand `example.txt'  
Try `tr --help' for more information.  
[sysadmin@localhost ~]$
```

Redirection de l'entrée standard (STDIN)

- **exemple(suite)** :ici **tr** lit les données à partir du fichier **example.txt** au lieu de les lire à partir du clavier on a donc rediriger l'entrée standard de la commande tr

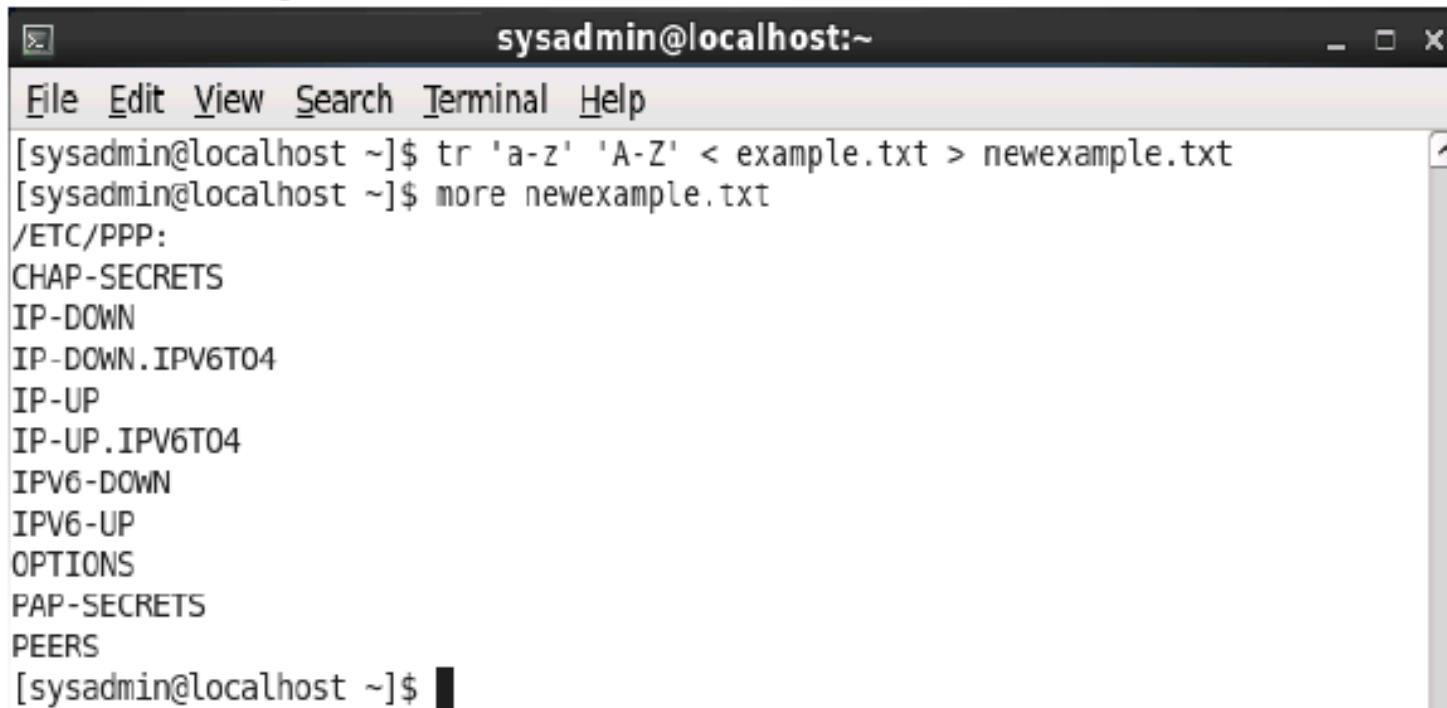


```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ tr 'a-z' 'A-Z' < example.txt  
/ETC/PPP:  
CHAP-SECRETS  
IP-DOWN  
IP-DOWN.IPV6T04  
IP-UP  
IP-UP.IPV6T04  
IPV6-DOWN  
IPV6-UP  
OPTIONS  
PAP-SECRETS  
PEERS  
[sysadmin@localhost ~]$
```

- Remarque : la redirection de l'entrée standard est souvent utilisée pour des commande qui n'accepte pas un nom de fichier comme argument comme le cas de tr

Redirection de l'entrée standard (STDIN)

- Exemple : cette exemple montre qu'on peut utiliser pour la même commande la redirection de STDIN et STDOUT
- tr lit à partir d'un fichier example.txt et écrit dans un autre fichier newexample.txt



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ tr 'a-z' 'A-Z' < example.txt > newexample.txt  
[sysadmin@localhost ~]$ more newexample.txt  
/ETC/PPP:  
CHAP-SECRETS  
IP-DOWN  
IP-DOWN.IPV6T04  
IP-UP  
IP-UP.IPV6T04  
IPV6-DOWN  
IPV6-UP  
OPTIONS  
PAP-SECRETS  
PEERS  
[sysadmin@localhost ~]$
```

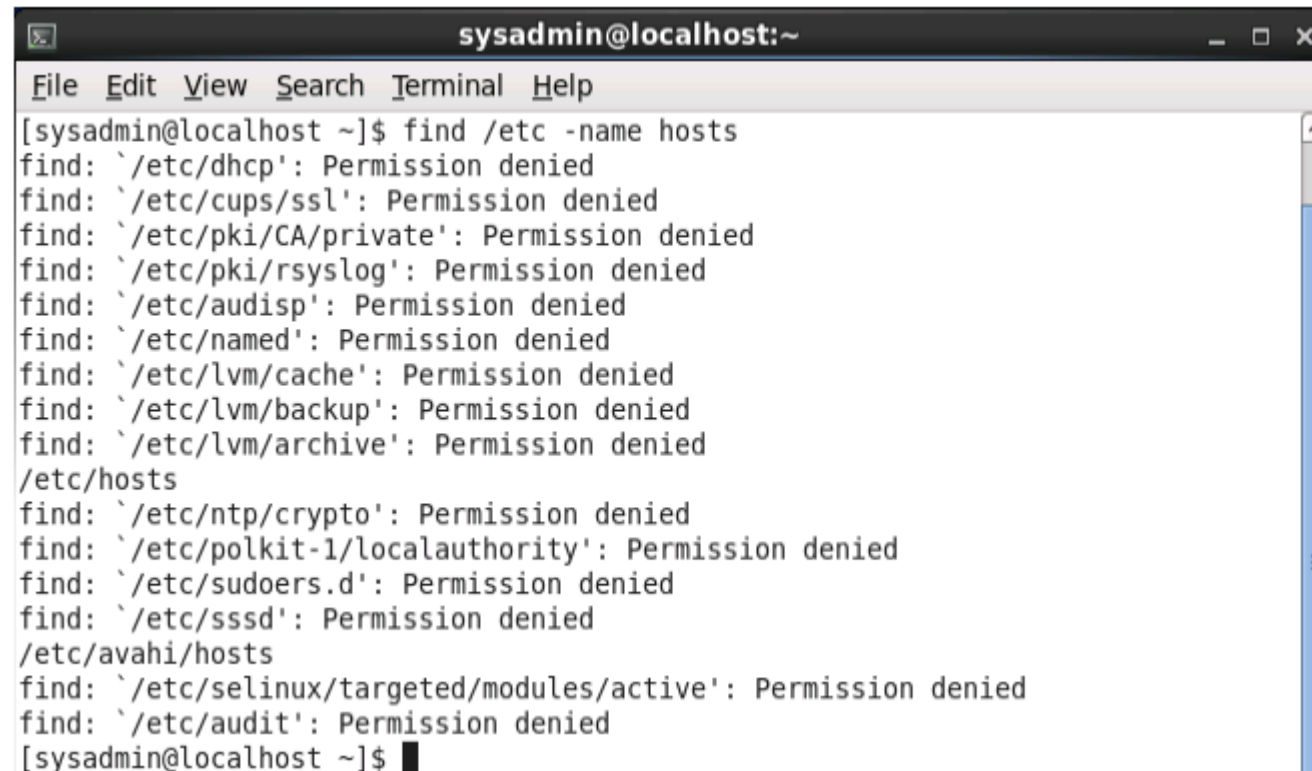
L'ordre des redirections compte ici il faut commence par l'entrée puis la sortie.

Recherche de fichier avec find

- La commande **find** est un outil puissant permettant de rechercher les fichiers sur le système de fichiers Linux , avec **find** on peut chercher les fichiers par nom , par taille , par type , par propriétaire...
- La syntaxe de la commande **find** est la suivante:
find [répertoire de départ] [options de recherche] [critères de recherche] [action]
- **Répertoire de départ** : indique l'emplacement où la recherche sera effectuée , find va chercher dans ce répertoire et dans tous ses sous-répertoires , si cette option n'est pas spécifié le répertoire courant sera l'emplacement de recherche par défaut.
- **Options de recherche** : indique les méta-data sur lesquelles on va se baser pour rechercher le fichier (nom , date , taille...)
- **Critères de recherche** : associés à l'option de recherche spécifient sur quels critères on va chercher le fichier (si on cherche par nom ça sera le nom du fichier qui est donné comme critère)
- **Action** : indique quelle est l'action à exécuter une fois le fichiers trouvé, si aucune action n'est spécifiée le nom du fichier sera affiché à l'écran

Recherche de fichier avec find : recherche par nom

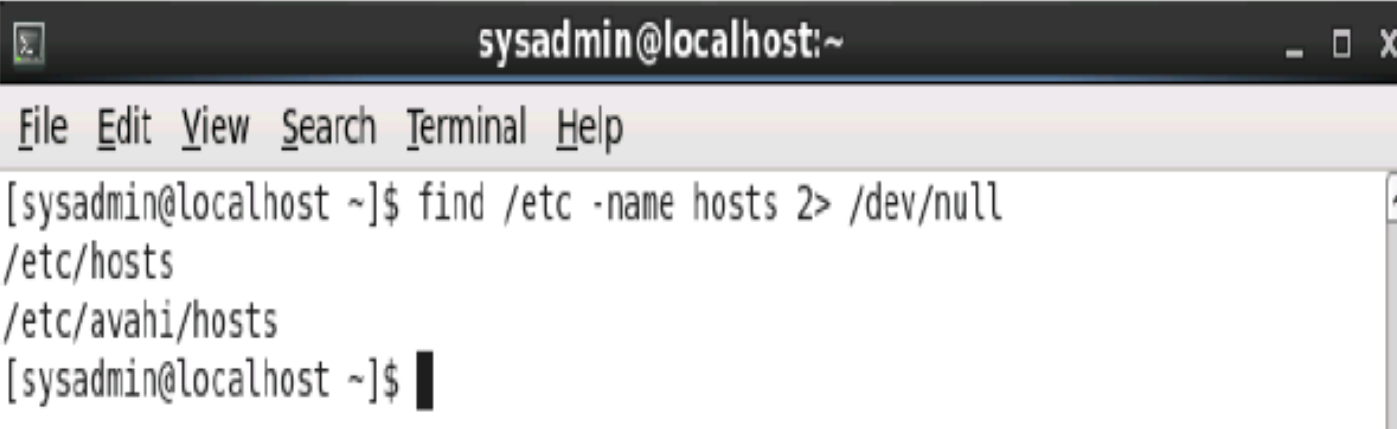
- Pour chercher des fichiers par nom on utilise l'option **-name** de la commande **find**
- Exemple 1 : ici deux fichiers portant comme nom hosts ont été trouvé (/etc/hosts et /etc/avahi/hosts) le reste c'est des messages d'erreur car l'utilisateur ayant lancé la commande n'a pas assez de droit



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ find /etc -name hosts  
find: `/etc/dhcp': Permission denied  
find: `/etc/cups/ssl': Permission denied  
find: `/etc/pki/CA/private': Permission denied  
find: `/etc/pki/rsyslog': Permission denied  
find: `/etc/audisp': Permission denied  
find: `/etc/named': Permission denied  
find: `/etc/lvm/cache': Permission denied  
find: `/etc/lvm/backup': Permission denied  
find: `/etc/lvm/archive': Permission denied  
/etc/hosts  
find: `/etc/ntp/crypto': Permission denied  
find: `/etc/polkit-1/localauthority': Permission denied  
find: `/etc/sudoers.d': Permission denied  
find: `/etc/sssd': Permission denied  
/etc/avahi/hosts  
find: `/etc/selinux/targeted/modules/active': Permission denied  
find: `/etc/audit': Permission denied  
[sysadmin@localhost ~]$
```

Recherche de fichier avec find : recherche par nom

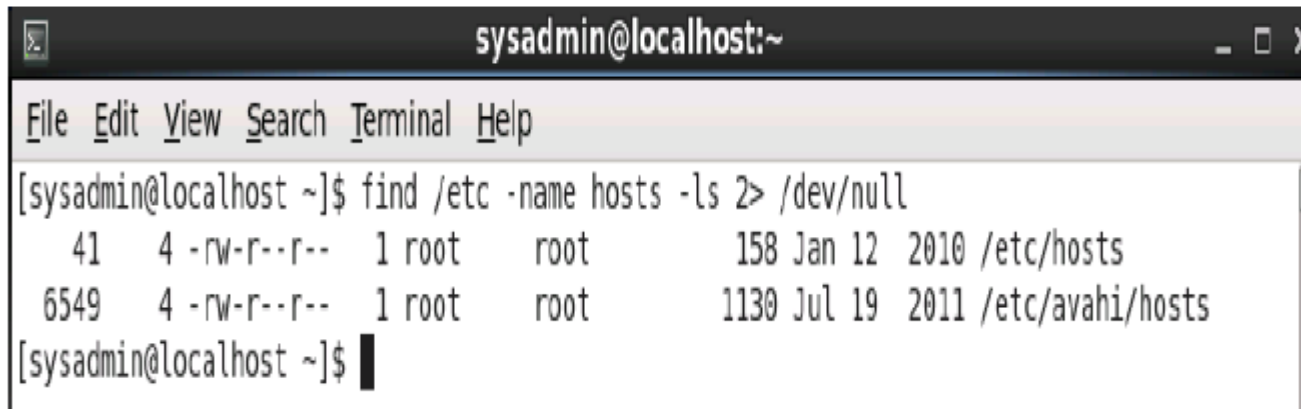
- Exemple 1(suite) : Pour rendre la sortie de la commande précédente plus lisible on peut rediriger le flux d'erreur standard(STDERR) vers le fichier /dev/null, /dev/null est utilisé pour rediriger tout flux indésirable car tout ce qui est redirigé vers /dev/null sera supprimé et ignoré.



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ find /etc -name hosts 2> /dev/null  
/etc/hosts  
/etc/avahi/hosts  
[sysadmin@localhost ~]$
```

Recherche de fichier avec find : recherche par nom

- Parfois il serait intéressant de chercher les fichiers et d'afficher les informations détaillées sur ces derniers ce qui va vous permettre de trouver plus facilement le fichier recherché
- Pour afficher ces informations on utilise l'option -ls de find
- Dans cet affichage la première colonne représente le numéro d'inode et la deuxième le nombre de bloc occupé par le fichier sur le disque les autres colonnes représentent les informations que fournit la commande ls -l



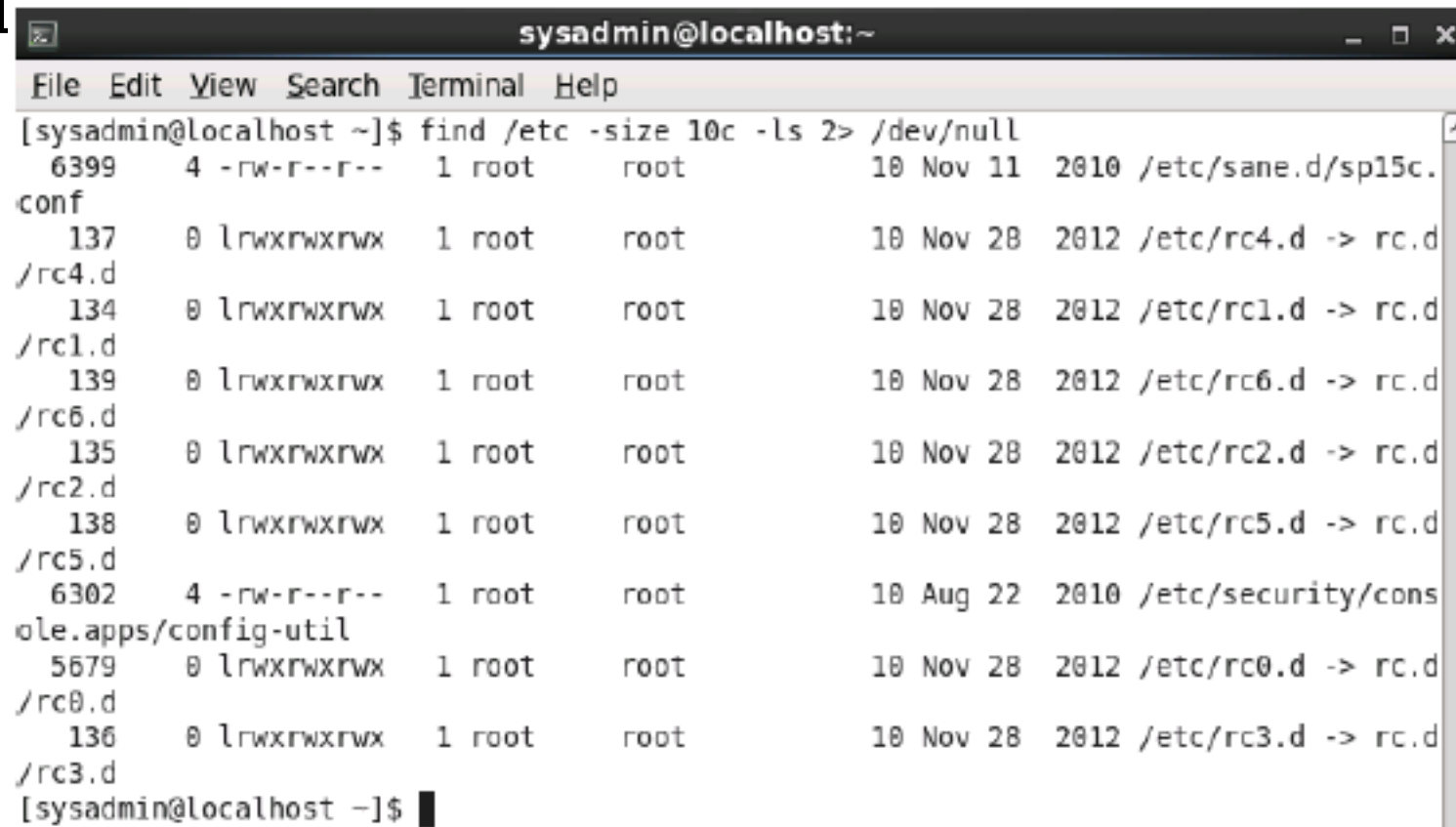
```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ find /etc -name hosts -ls 2> /dev/null  
  41    4 -rw-r--r--  1 root   root      158 Jan 12  2010 /etc/hosts  
6549    4 -rw-r--r--  1 root   root     1130 Jul 19  2011 /etc/avahi/hosts  
[sysadmin@localhost ~]$
```

Recherche de fichier avec find : recherche par taille

- L'option **-size** de **find** permet de chercher les fichiers par taille ,cette option permet de chercher les fichiers supérieur à une taille donnée , inférieur à une taille donnée ou qui exactement une taille donnée
- Lorsqu'on utilise l'option -size on peut spécifié la taille des fichiers à rechercher par octets(c),par kilooctets(k),par mégaoctets(M) ou par giga-octets (G)

Recherche de fichier avec find : recherche par taille

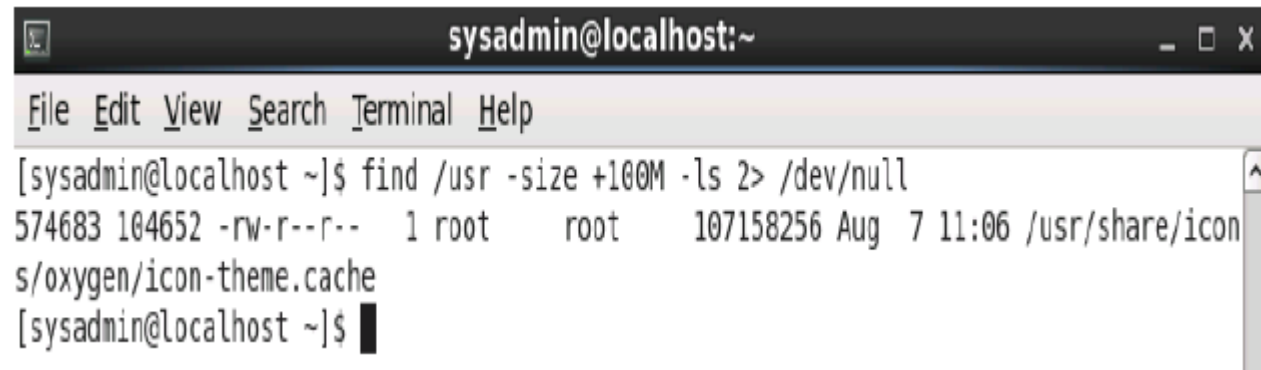
- Exemple1 : ici on cherche tous les fichiers contenus dans /etc et qui ont exactement 10 octets



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ find /etc -size 10c -ls 2> /dev/null  
 6399   4 -rw-r--r--   1 root    root      10 Nov 11  2010 /etc/sane.d/sp15c.  
conf  
   137   0 lrwxrwxrwx   1 root    root      10 Nov 28  2012 /etc/rc4.d -> rc.d  
/rc4.d  
   134   0 lrwxrwxrwx   1 root    root      10 Nov 28  2012 /etc/rc1.d -> rc.d  
/rc1.d  
   139   0 lrwxrwxrwx   1 root    root      10 Nov 28  2012 /etc/rc6.d -> rc.d  
/rc6.d  
   135   0 lrwxrwxrwx   1 root    root      10 Nov 28  2012 /etc/rc2.d -> rc.d  
/rc2.d  
   138   0 lrwxrwxrwx   1 root    root      10 Nov 28  2012 /etc/rc5.d -> rc.d  
/rc5.d  
 6302   4 -rw-r--r--   1 root    root     10 Aug 22  2010 /etc/security/cons  
ole.apps/config-util  
 5679   0 lrwxrwxrwx   1 root    root      10 Nov 28  2012 /etc/rc0.d -> rc.d  
/rc0.d  
   136   0 lrwxrwxrwx   1 root    root      10 Nov 28  2012 /etc/rc3.d -> rc.d  
/rc3.d  
[sysadmin@localhost ~]$
```

- **Exemple 2** : si on veut chercher les fichiers qui ont une taille supérieur à une valeur donnée en place le caractère “+” avant la taille.

ici on cherche dans /etc tous les fichiers qui ont une taille supérieur à 100Mo



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ find /usr -size +100M -ls 2> /dev/null  
574683 104652 -rw-r--r-- 1 root    root    107158256 Aug  7 11:06 /usr/share/icon  
s/oxygen/icon-theme.cache  
[sysadmin@localhost ~]$
```

Pour chercher les fichiers ayant une taille inférieur à une valeur donnée on place le caractère “-” avant la taille

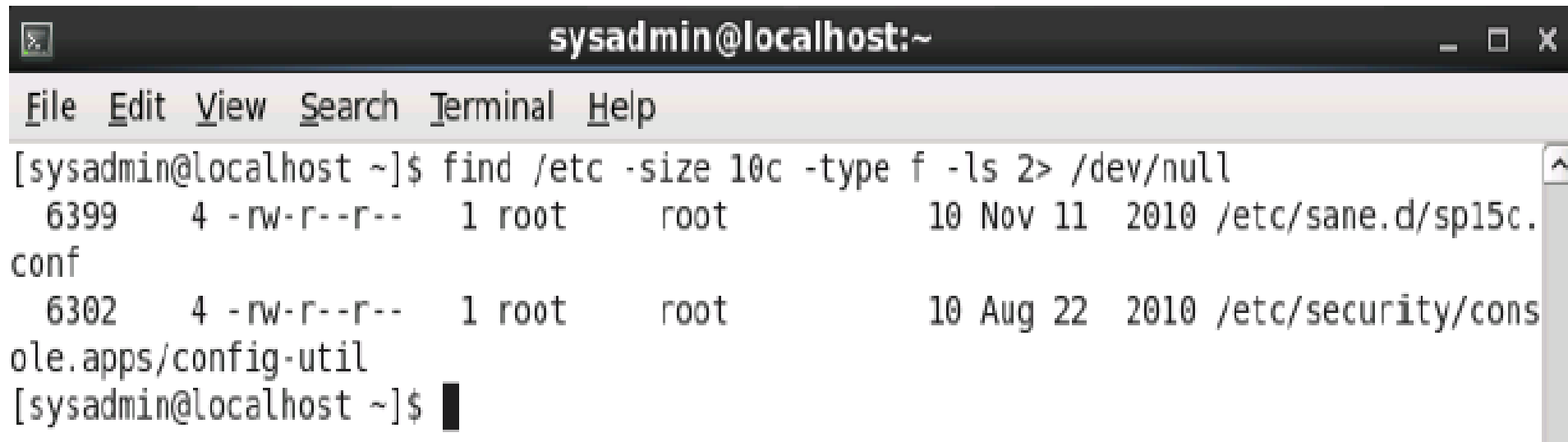
Recherche de fichier avec find : autres options de find

– Find possède plusieurs options de recherche le tableau suivant illustre les plus importantes:

<i>option</i>	<i>Description</i>
-maxdepth	Indique la profondeur de recherche dans l'arborescence des répertoires, par exemple -maxdepth 1 indique à la commande de chercher dans le répertoire et dans ces sous-répertoires immédiats.
-group	Permet de chercher les fichiers appartenant à un groupe donné, par exemple -group tech cherche les fichiers appartenant au groupe tech.
-iname	Retourne les fichiers qui correspondent au nom donné comme argument cette option n'est pas sensible à la casse comme -name, par exemple -iname hosts chercher tous les fichiers dont les noms peuvent être hosts, Hosts, HOSTS...
-mmin	Cherche les fichiers en se basant sur la date de modification en minutes, par exemple -mmin 10 retourne les fichiers qui ont été modifiés il y a 10 minutes.
-type	Cherche les fichiers correspondant à un type donné, par exemple -type f retourne les fichiers normaux -type d retourne les fichiers de type répertoire.
-user	Cherche les fichiers appartenant à un utilisateur, par exemple -user bob cherche les fichiers dont l'utilisateur bob est propriétaire.

Recherche de fichier avec find : combiner les options de find

- On peut combiner plusieurs options de **find** dans ce cas ces options agissent comme un AND logique c'est-à-dire les fichiers retournés doivent satisfaire tous les critères
- **Exemple** : on veut chercher les fichiers réguliers(normaux) se trouvant dans **/etc** et qui ont exactement **10 octets** de taille



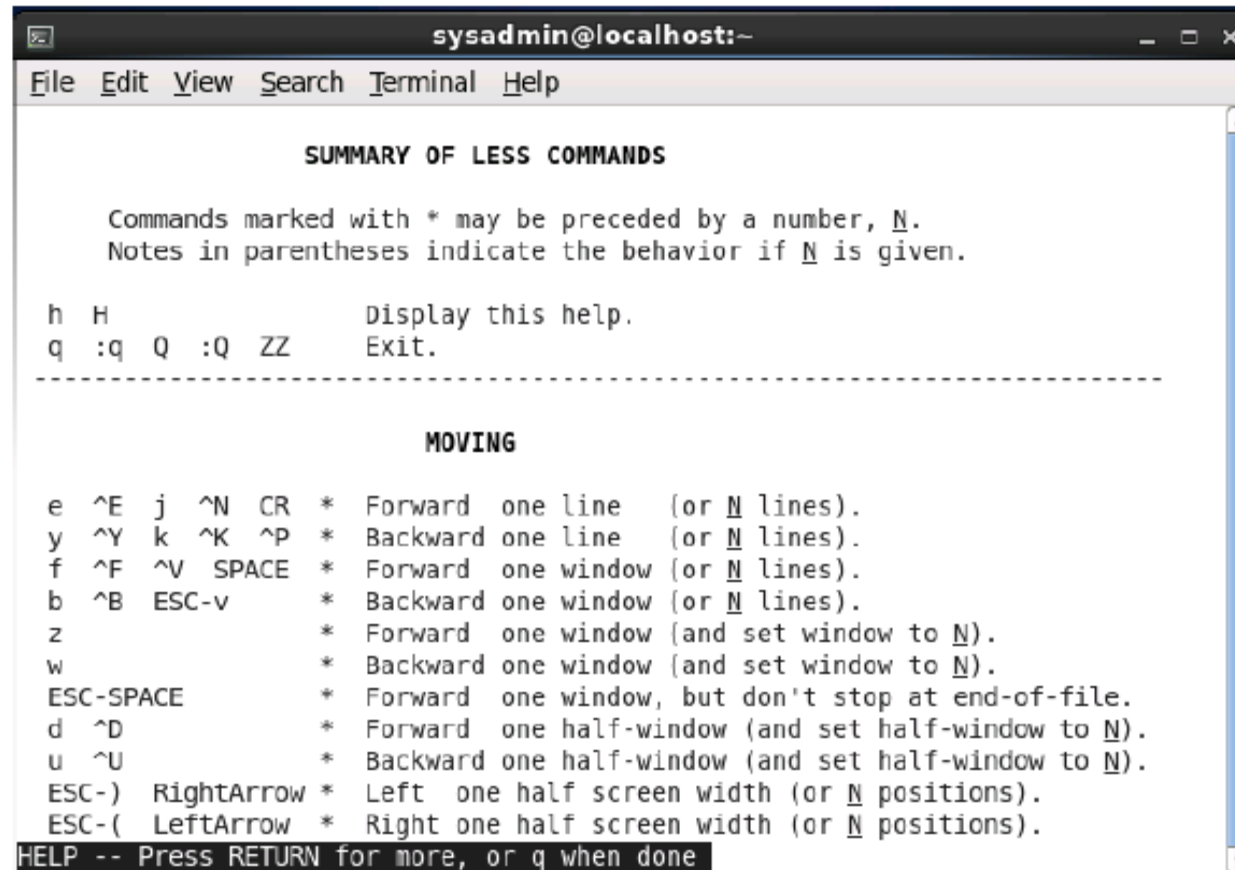
```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ find /etc -size 10c -type f -ls 2> /dev/null  
  6399    4 -rw-r--r--    1 root    root      10 Nov 11  2010 /etc/sane.d/sp15c.  
conf  
  6302    4 -rw-r--r--    1 root    root      10 Aug 22  2010 /etc/security/cons  
ole.apps/config-util  
[sysadmin@localhost ~]$
```

Affichage du contenu des fichiers avec less

- La commande **cat** permet d'afficher le contenu des fichiers texte elle ne pose pas de problème pour les petits fichiers mais elle reste un mauvais choix pour afficher le contenu des fichiers larges
- Pour afficher les fichiers large on aura besoin de commande qui permet d'afficher le fichier page par page et d'offrir aussi la possibilité de naviguer à travers le fichier en utilisant les touches du clavier , pour ces raisons on utilise deux commandes:
- **less** : cette commande offre plusieurs fonctionnalités pour afficher les fichiers , c'est la commande utilisée pour afficher *les pages man*.
- **more** : possède moins de fonctionnalités que less , puisque less est basé sur more les deux commandes utilisent les mêmes raccourcis clavier pour naviguer dans un fichier.

Affichage du contenu des fichiers avec less : affichage de l'aide de less

- Lorsqu'on affiche le contenu d'un fichier avec less si on appuis sur la touche h du clavier une aide sera affichée montrant tous les raccourcis claviers qu'on peut utiliser avec less



```
sysadmin@localhost:~
File Edit View Search Terminal Help

SUMMARY OF LESS COMMANDS

Commands marked with * may be preceded by a number, N.
Notes in parentheses indicate the behavior if N is given.

h H          Display this help.
q :q Q :Q ZZ  Exit.
-----

MOVING

e ^E j ^N CR * Forward one line (or N lines).
y ^Y k ^K ^P * Backward one line (or N lines).
f ^F ^V SPACE * Forward one window (or N lines).
b ^B ESC-v    * Backward one window (or N lines).
z              * Forward one window (and set window to N).
w              * Backward one window (and set window to N).
ESC-SPACE     * Forward one window, but don't stop at end-of-file.
d ^D          * Forward one half-window (and set half-window to N).
u ^U          * Backward one half-window (and set half-window to N).
ESC-) RightArrow * Left one half screen width (or N positions).
ESC-( LeftArrow  * Right one half screen width (or N positions).

HELP -- Press RETURN for more, or q when done
```

Affichage du contenu des fichiers avec less : naviguer dans le contenu d'un fichier

- Pour naviguer à travers le contenu d'un fichier on utilise des touches claviers (dont on peut visualiser l'aide avec la touche h)
- La plupart de ses touches sont similaires pour **less** et **more** le ,tableau suivant résume les plus importantes:

Mouvement	Touche clavier
Page suivante	espace
Page précédente	b
Ligne suivante	entrée
quitter	q
aide	h

Affichage du contenu des fichiers avec less:chercher dans le contenu d'un fichier

- Pour chercher dans affichage de less à partir de notre emplacement vers la fin du fichier on utilise la touche “/” suivie du terme à rechercher puis on tape entrée toutes les correspondance trouvées seront mise en surbrillance
- Pour chercher dans affichage de less à partir de notre emplacement vers le début du fichier on utilise la touche “?” suivie du terme à rechercher puis on tape entrée toutes les correspondance trouvées seront mise en surbrillance pour naviguer entre les termes trouvées on utilise les touches “n” (aller au suivant) et “N” (aller au précédent)

Affichage du contenu des fichiers avec less : chercher dans le contenu d'un fichier

- **exemple**



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
Afrogaia  
Afrogaian  
afront  
afroformosa  
afros  
Afro-semitic  
afrown  
AFS  
AFSC  
AFSCME  
Afshah  
Afshar  
AFSK  
AFT  
aft  
aftaba  
after  
after-  
after-acquired  
afteract  
afterage  
afterattack  
afterband  
:  
█
```

Les commandes head et tail

- Comme on a vu précédemment les commandes **head** et **tail** permettent de filtrer un fichier et d'en afficher un nombre limité de lignes
 - Pour afficher les n premières lignes d'un fichier on utilise **head**
 - Pour afficher les n dernières lignes d'un fichier on utilise **tail**
- Par défaut les deux commandes affiche 10 lignes
- Le tableau suivant donne des exemples de l'utilisation de head et tail.
- Les commandes head et tail peuvent aussi faire un affichage en se basant sur le nombre d'octets pour cela il faut utiliser l'option -c au lieu de -n
- Les options -n et -c sont utilisée de la même façon

Les commandes head et tail

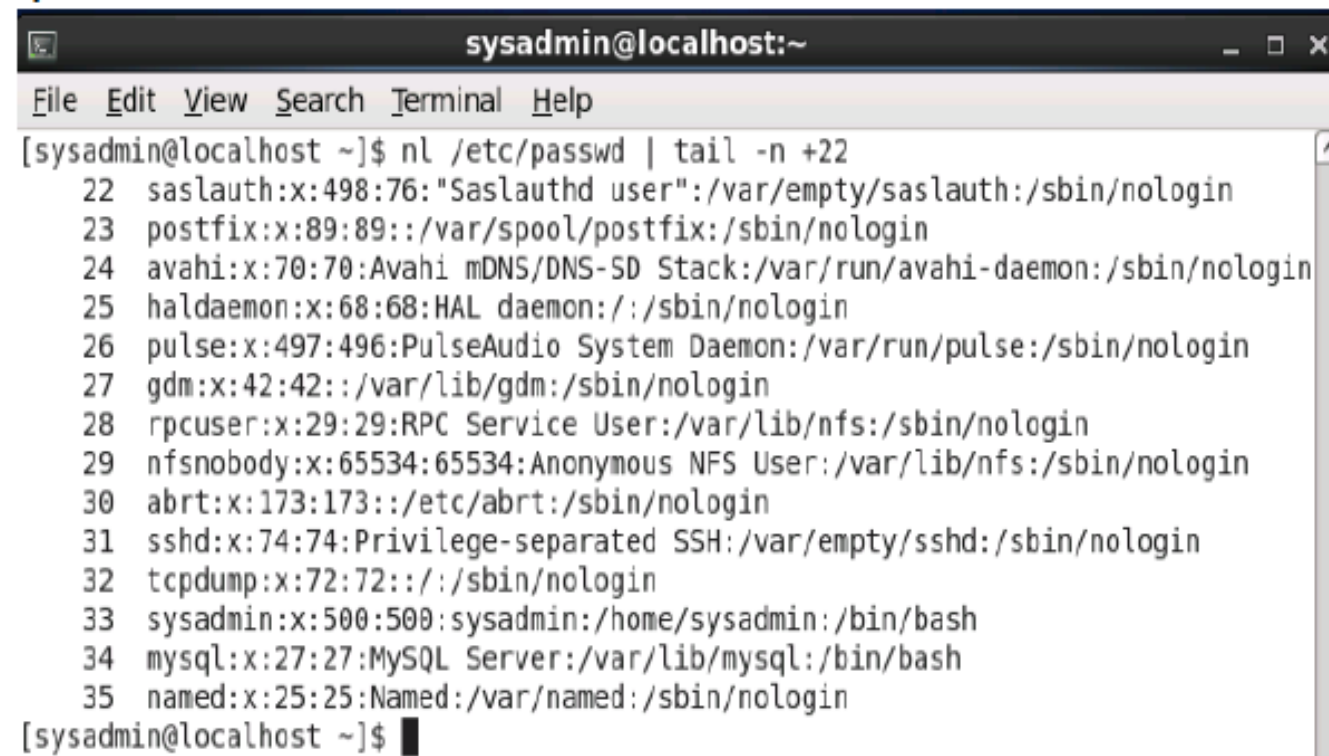
exemple	Description
<code>head /etc/passwd</code>	Les 10 premières lignes du fichier <code>/etc/passwd</code>
<code>head -3 /etc/group</code>	Les 3 premières lignes du fichier <code>/etc/group</code>
<code>head -n 3 /etc/group</code>	Les 3 premières lignes du fichier <code>/etc/group</code>
<code>help head</code>	Les 10 premières lignes de la sortie de la commande <code>help</code>
<code>head -c 5 /etc/passwd</code>	Les 5 premier octets du fichier <code>/etc/passwd</code>
<code>tail /etc/group</code>	Les dix dernières lignes du fichier <code>/etc/group</code>
<code>tail -5 /etc/passwd</code>	Les 5 dernières lignes du fichier <code>/etc/passwd</code>
<code>tail -n 5 /etc/passwd</code>	Les 5 dernières lignes de du fichier <code>/etc/passwd</code>
<code>tail -c 5 /etc/passwd</code>	Les 5 dernier octets du fichier <code>/etc/passwd</code>
<code>help tail</code>	Les 10 dernières lignes de la sortie de la commande <code>help</code>

Les commandes head et tail

- Pour la commande head on peut afficher le nombre de lignes on donnant comme option: -n nombre ou -nombre
- Exemple:
 - head -n 3 /etc/passwd et head -3 /etc/passwd donneront le même résultat (afficher les 3 premières lignes du fichier /etc/passwd)
- Mais lorsqu'on donne comme option -n -nombre on demande à la commande head d'afficher toutes les lignes sauf les dernières lignes indiquées par nombre
- Exemple:
 - head -n -3 /etc/passwd indique à head d'afficher toutes les lignes de /etc/passwd sauf les 3 dernières

Les commandes head et tail

- Pour la commande tail si on utilise une valeur positive(+nbr) avec l'option -n ça permet d'afficher les dernières lignes à partir de la ligne numéro nbr
- Exemple :



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ nl /etc/passwd | tail -n +22  
22 saslauth:x:498:76:"Saslauthd user":/var/empty/saslauth:/sbin/nologin  
23 postfix:x:89:89::/var/spool/postfix:/sbin/nologin  
24 avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon:/sbin/nologin  
25 haldaemon:x:68:68:HAL daemon:/:/sbin/nologin  
26 pulse:x:497:496:PulseAudio System Daemon:/var/run/pulse:/sbin/nologin  
27 gdm:x:42:42::/var/lib/gdm:/sbin/nologin  
28 rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin  
29 nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin  
30 abrt:x:173:173::/etc/abrt:/sbin/nologin  
31 sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin  
32 tcpdump:x:72:72::/:/sbin/nologin  
33 sysadmin:x:500:500:sysadmin:/home/sysadmin:/bin/bash  
34 mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash  
35 named:x:25:25:Named:/var/named:/sbin/nologin  
[sysadmin@localhost ~]$
```

Tri d'un fichier en utilisant la commande sort

- La commande sort permet de trier les lignes d'un fichier par ordre alphanumérique ou par ordre de valeur en se basant sur plusieurs champs contenus de la ligne , ces champs sont séparés par un séparateur qui peut être par exemple un espace.
- Exemple 1:
 - on va créer un fichier contenant 5 lignes à l'aide de la commande head

```
sysadmin@localhost:~$ head -5 /etc/passwd > mypasswd  
sysadmin@localhost:~$
```

```
sysadmin@localhost:~$ cat mypasswd  
root:x:0:0:root:/root:/bin/bash  
daemon:x:1:1:daemon:/usr/sbin:/bin/sh  
bin:x:2:2:bin:/bin:/bin/sh  
sys:x:3:3:sys:/dev:/bin/sh  
sync:x:4:65534:sync:/bin:/bin/sync  
sysadmin@localhost:~$
```

Tri d'un fichier en utilisant la commande sort

- exemple1(suite)
 - Maintenant on va trier les lignes du fichier mypasswd en utilisant sort

```
sysadmin@localhost:~$ sort mypasswd
bin:x:2:2:bin:/bin:/bin/sh
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
root:x:0:0:root:/root:/bin/bash
sync:x:4:65534:sync:/bin:/bin/sync
sys:x:3:3:sys:/dev:/bin/sh
sysadmin@localhost:~$
```

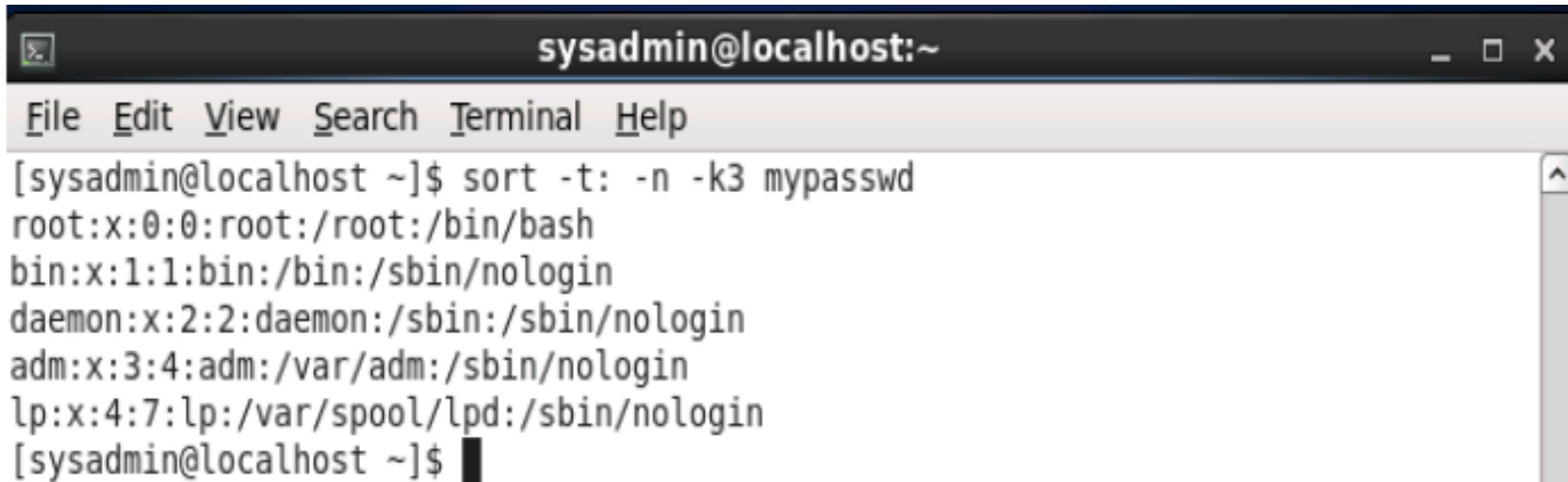
Tri d'un fichier en utilisant la commande sort

- Dans le cas où les champs d'une ligne d'un fichier sont séparés par un autre caractère séparateur(par exemple ; ou :) différent d'espace il faut utiliser l'option **-t** pour spécifier le séparateur.
- Pour spécifier le champ suivant lequel on va trier on utilise l'option **-k** suivie d'un argument indiquant le numéro du champ(le premier champ ayant le numéro 1)
- L'option **-n** permet de faire un tri numérique des champs
- L'option **-r** permet de faire un tri par ordre inverse(décroissant)

```
[sysadmin@localhost Bureau]$ sort fichier1
11
12
123
13
7
[sysadmin@localhost Bureau]$ sort -n fichier1
7
11
12
13
123
[sysadmin@localhost Bureau]$
```

Tri d'un fichier en utilisant la commande sort

- Exemple 2 : dans cet exemple on utilise :
 - l'option **-t** avec l'argument **:** pour indiquer que les champs de la ligne sont séparés par **:**
 - L'option **-n** pour faire un tri par valeur numérique
 - L'option **-k** avec l'argument **3** pour tri les lignes suivant la valeur du troisième champ
 - On peut utiliser l'option **-r** pour inverser l'ordre du tri



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ sort -t: -n -k3 mypasswd  
root:x:0:0:root:/root:/bin/bash  
bin:x:1:1:bin:/bin:/sbin/nologin  
daemon:x:2:2:daemon:/sbin:/sbin/nologin  
adm:x:3:4:adm:/var/adm:/sbin/nologin  
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin  
[sysadmin@localhost ~]$
```

Tri d'un fichier en utilisant la commande sort

- On peut faire un tri suivant les valeurs de plusieurs champs on prend l'exemple suivant
- Exemple 3:
 - Soit le fichier “personnes” contenant les données suivantes:
said:alaoui:67
rachid:alaoui:56
said:elouali:23
said:elouali:12
 - On veut trier les personnes contenues dans ce fichier par nom puis par prénom ensuite par age la commande permettant de réaliser cela est:
– **sort -t: -k2 -k1 -k3n personnes**

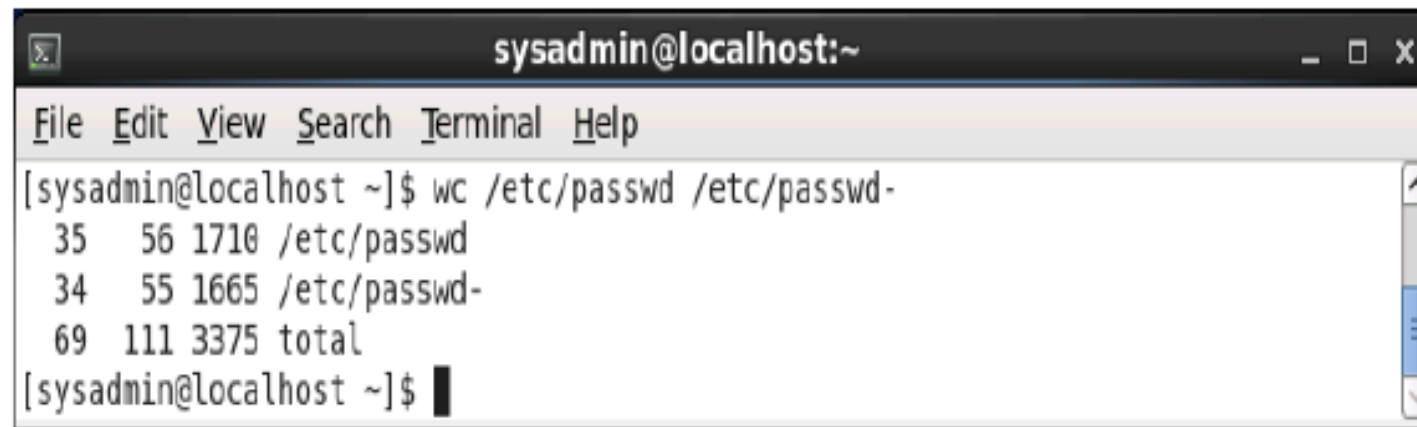
Tri d'un fichier en utilisant la commande sort

- Exemple 4 : trier un fichier contenant des adresses IP

```
sysadmin@localhost:~  
Fichier  Édition  Affichage  Rechercher  Terminal  Aide  
[sysadmin@localhost ~]$ cat adresses  
192.168.5.1  
192.168.5.15  
192.168.1.255  
192.168.1.1  
172.16.0.1  
172.16.9.1  
192.168.1.20  
192.168.1.254  
172.16.255.0  
[sysadmin@localhost ~]$ sort -t. -k1n -k2n -k3n -k4n adresses  
172.16.0.1  
172.16.9.1  
172.16.255.0  
192.168.1.1  
192.168.1.20  
192.168.1.254  
192.168.1.255  
192.168.5.1  
192.168.5.15  
[sysadmin@localhost ~]$
```


Afficher les statistiques d'un fichier avec wc

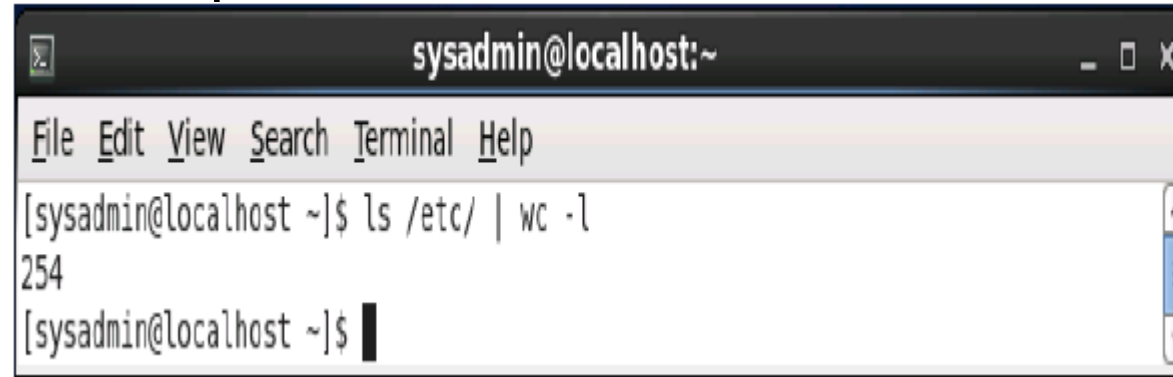
- La commande **wc** permet d'afficher trois types de statistiques concernant les fichiers passés comme argument : le nombre de ligne , le nombre de mots et le nombre de caractères (***1caractère=1octet***)
- Exemple1:ici on affiche les statistiques pour 2 fichiers /etc/passwd et /etc/passwd-,le résultat contient 4 colonnes (nombre de lignes , nombre de mots , nombre d'octets et le nom du fichier)



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ wc /etc/passwd /etc/passwd-  
 35   56 1710 /etc/passwd  
 34   55 1665 /etc/passwd-  
 69  111 3375 total  
[sysadmin@localhost ~]$
```

Afficher les statistiques d'un fichier avec wc

- – Pour afficher uniquement des statistiques spécifiques on utilise les options suivantes de **wc** -l afficher le nombre de lignes uniquement, -w afficher le nombre de mots uniquement et -c afficher le nombre de caractères uniquement
- – La commande **wc** peut être utilisée pour compter le nombre de lignes que comporte le résultat d'une commande à travers un pipe, par exemple on souhaite compter le nombre de fichier que contient le répertoire /etc, il faut exécuter **ls /etc** puis faire passer son résultat à **wc -l**



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ ls /etc/ | wc -l  
254  
[sysadmin@localhost ~]$
```

Filtrer le contenu d'un fichier avec cut

- La commande **cut** permet d'extraire des colonnes à partir d'un fichier ou à partir de l'entrée standard, la commande **cut** agit sur les fichiers contenant un délimiteur qui est par défaut espace (la même chose que la commande sort)
- L'option **-d** de cut permet de spécifier un délimiteur différent d'espace comme point virgule, point...
- L'option **-f *num*** permet de spécifier les champs à extraire, le premier champ étant numéro **1**, ces numéros de champs peuvent être donnés sous forme de valeurs séparées par des virgules ou sous forme d'intervalle.

Exemple de la liste de numéro de champ (colonne) ***num***

3-9 : de 3 à 9

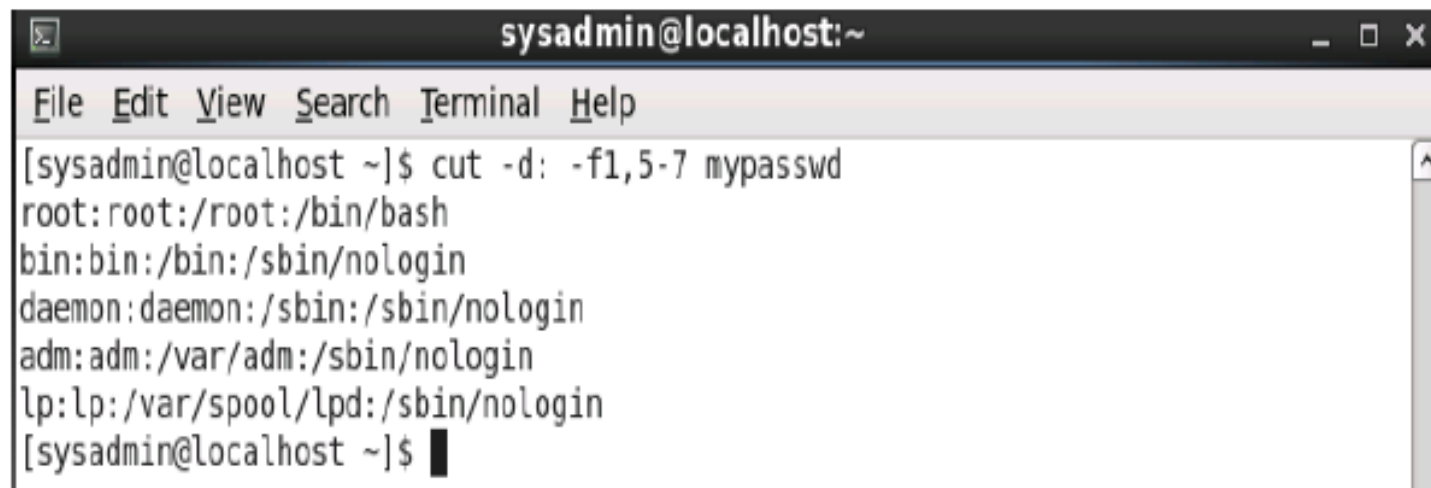
1,3,5 : 1, 3 et 5

-5 : de 1 à 5

5- : de 5 à la fin de la ligne

Filtrer le contenu d'un fichier avec cut

- Exemple 1: dans cet exemple le premier , le 5ème, le 6ème et le 7ème champ du fichier mypasswd seront affichés

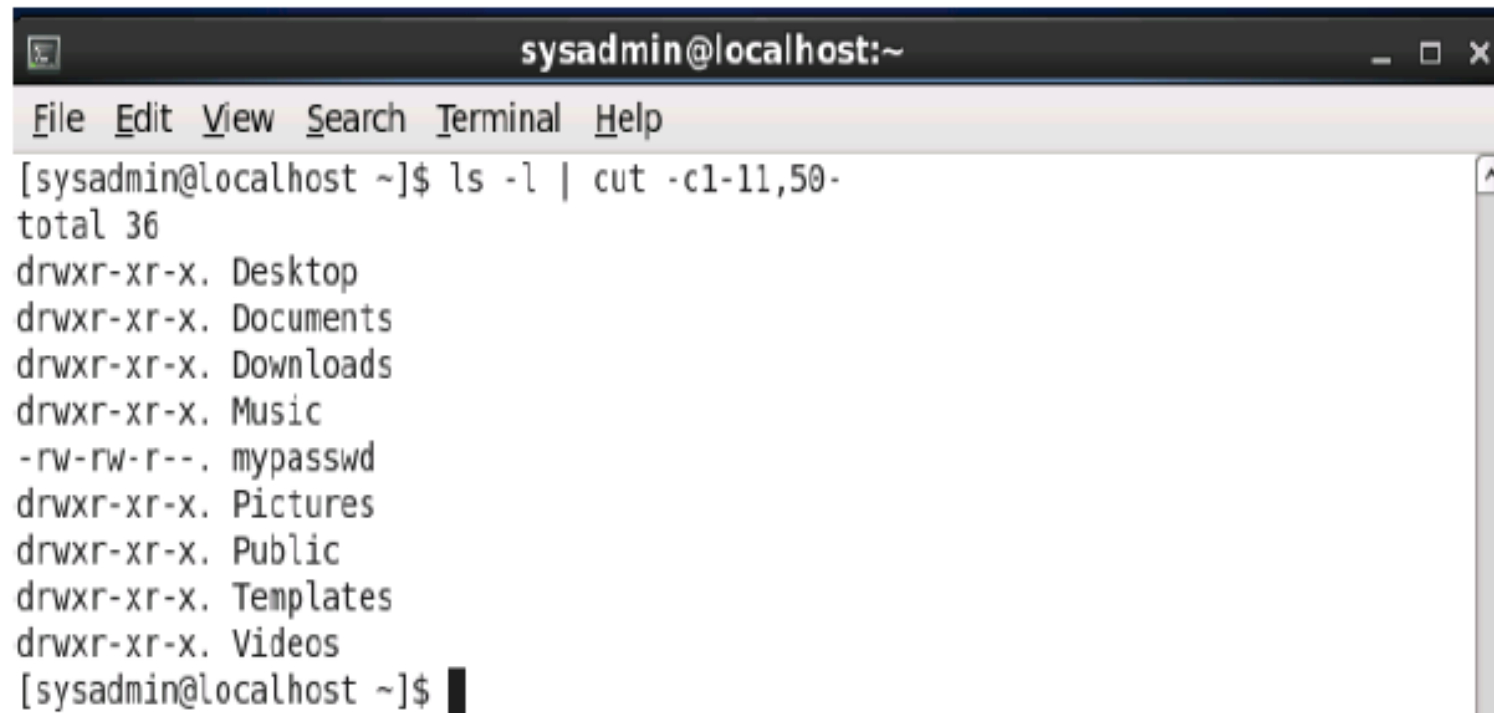


```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ cut -d: -f1,5-7 mypasswd  
root:root:/root:/bin/bash  
bin:bin:/bin:/sbin/nologin  
daemon:daemon:/sbin:/sbin/nologin  
adm:adm:/var/adm:/sbin/nologin  
lp:lp:/var/spool/lpd:/sbin/nologin  
[sysadmin@localhost ~]$
```

- Avec cut on peut aussi extraire le contenu d'une ligne en se basant sur la position d'un caractère et non celle d'un champ pour cela il faut utiliser l'option **-c num** comme le montre l'exemple.

Filtrer le contenu d'un fichier avec cut

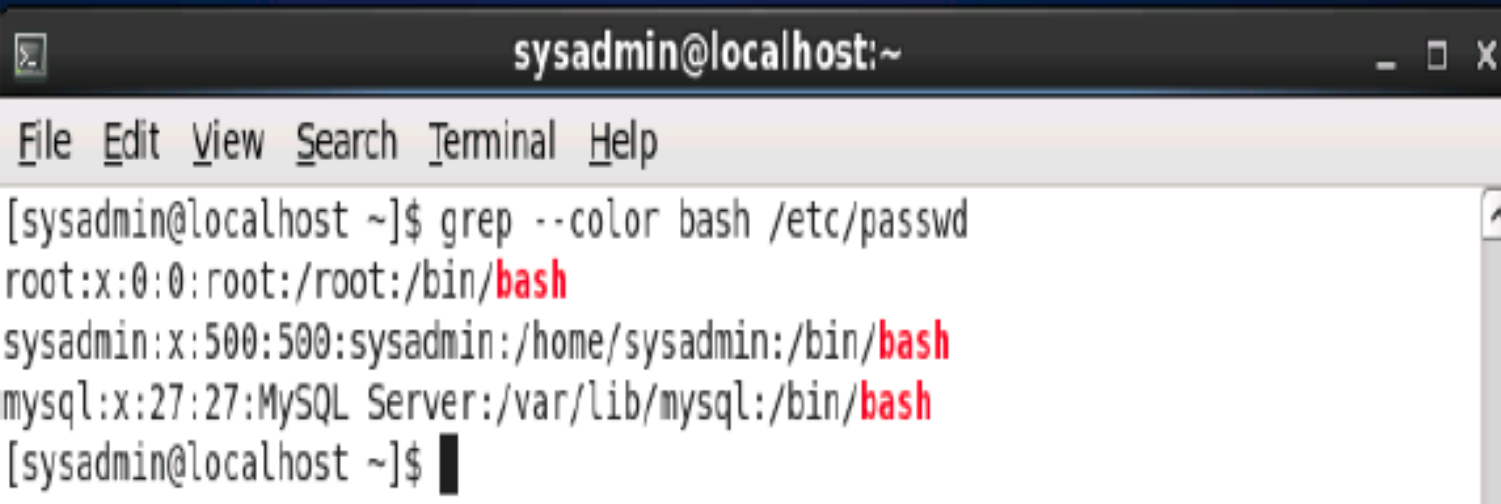
- Exemple 2 : on va extraire à partir du résultat de `ls -l` le type de fichier (caractère 1) les permissions(caractère 2-10) et le nom du fichier (caractère 50 jusqu'à la fin)



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ ls -l | cut -c1-11,50-  
total 36  
drwxr-xr-x. Desktop  
drwxr-xr-x. Documents  
drwxr-xr-x. Downloads  
drwxr-xr-x. Music  
-rw-rw-r--. mypasswd  
drwxr-xr-x. Pictures  
drwxr-xr-x. Public  
drwxr-xr-x. Templates  
drwxr-xr-x. Videos  
[sysadmin@localhost ~]$
```

Filtrer le contenu d'un fichier avec grep

- **grep** permet de filtrer le contenu d'un fichier en se basant sur les lignes correspondant à un terme donné , le terme cherché peut être un mot simple ou fourni à travers des expressions régulières.
- Exemple 1:à partir du fichier /etc/passwd on veut afficher(filtrer) les lignes contenant le terme bash

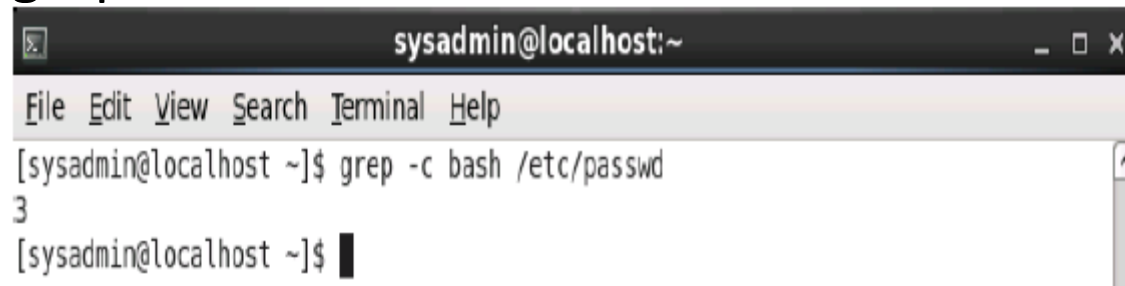


```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ grep --color bash /etc/passwd  
root:x:0:0:root:/root:/bin/bash  
sysadmin:x:500:500:sysadmin:/home/sysadmin:/bin/bash  
mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash  
[sysadmin@localhost ~]$
```

– L'option **--color** permet d'afficher le terme recherché en rouge , dans la plupart des distributions il existe un alias (**alias grep='grep --color'**) qui permet d'utiliser l'option **--color** automatiquement

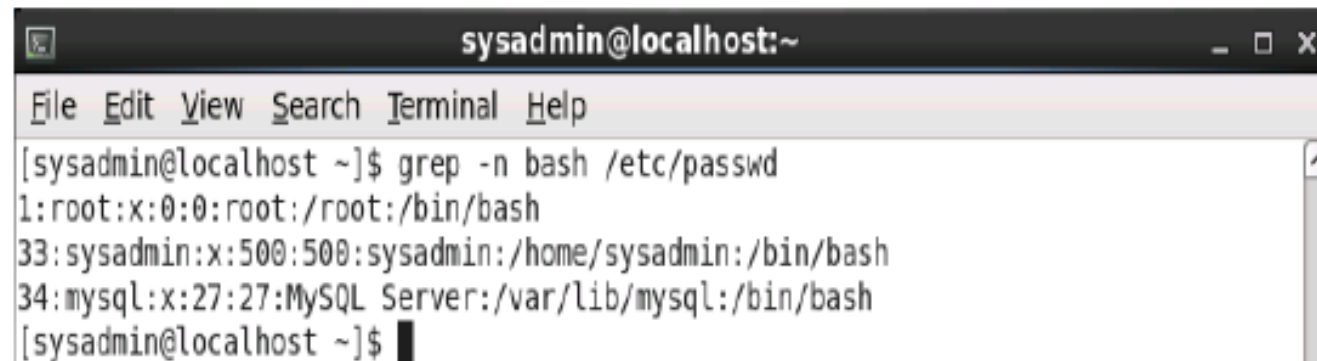
Filtrer le contenu d'un fichier avec grep

- Exemple2 : pour afficher combien de ligne correspondant au terme recherché on utilise l'option -c de grep



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ grep -c bash /etc/passwd  
3  
[sysadmin@localhost ~]$
```

- Exemple3 : pour afficher les numéros des lignes du fichier original correspondante au terme recherché on utilise l'option -n de **grep**



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ grep -n bash /etc/passwd  
1:root:x:0:0:root:/root:/bin/bash  
33:sysadmin:x:500:500:sysadmin:/home/sysadmin:/bin/bash  
34:mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash  
[sysadmin@localhost ~]$
```

Filtrer le contenu d'un fichier avec grep

– Autres exemple de grep

commande	résultat	Description de l'option
<code>grep -v nologin /etc/passwd</code>	Toutes les lignes qui ne contiennent pas « nologin » dans <code>/etc/passwd</code>	-v: permet de sélectionner les lignes qui ne correspondent pas
<code>grep -l linux /etc/*</code>	Afficher tous les fichiers contenu dans <code>/etc</code> et qui contiennent le terme linux	-l: affiche uniquement les fichiers où il ya correspondance
<code>grep -i linux /etc/*</code>	Afficher toutes les lignes de tous les fichiers de <code>/etc</code> contenant le terme linux (en majuscule ou en minuscule)	-i: ignorer la case
<code>grep -w linux /etc/*</code>	Afficher toutes les lignes de <code>/etc</code> contenant le mot entier linux.	-w: sélectionner les lignes contenant le mot entier

Les expressions régulières de base (BRE)

- Une expression régulière est une collection de caractères “**simples**” et “**spéciaux**” utilisée pour représenter un terme à chercher ou à filtrer avec une commande comme grep
- Dans une expression régulière un simple caractère ne représente que lui même (“r” représente “r”) tandis qu’un caractère spécial a une signification particulière pour le Shell
- Il existe les expressions régulières de base (BRE) qui peuvent être utilisées avec un nombre de commandes Linux et les expressions régulières étendues (ERE) qui sont elles utilisées avec les commandes Linux avancées

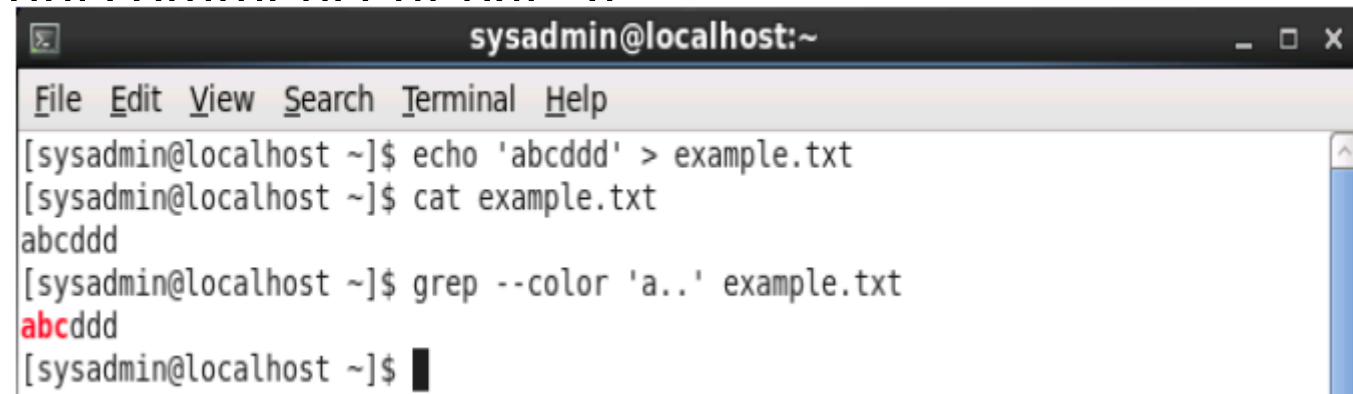
Les expressions régulières de base (BRE)

- Le tableau suivant représente quelques expressions régulières de base

Expression régulière	signification
.	Représente un caractère
[]	Un caractère dans une liste ou intervalle de caractères (si le premier caractère est « ^ » ça signifie tous caractère qui ne figure pas dans la liste
*	Caractère précédent répété 0 fois ou plus
^	Le texte qui suit doit figurer au début de la ligne
\$	Le texte qui précède doit figurer à la fin de la ligne

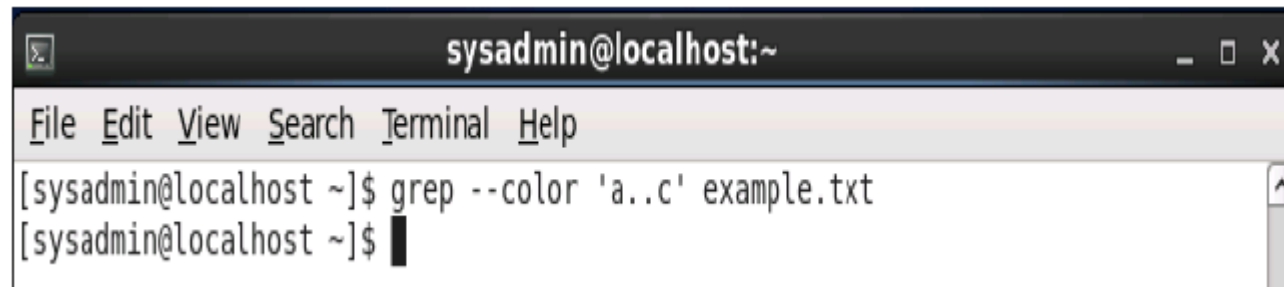
Les expressions régulières de base (bre):le caractère (.)

- **Exemple 1:**avec **grep** on cherche dans le fichier **example.txt** les termes correspondant à trois caractères qui commencent par "a"



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ echo 'abcd' > example.txt  
[sysadmin@localhost ~]$ cat example.txt  
abcd  
[sysadmin@localhost ~]$ grep --color 'a..' example.txt  
abcd  
[sysadmin@localhost ~]$
```

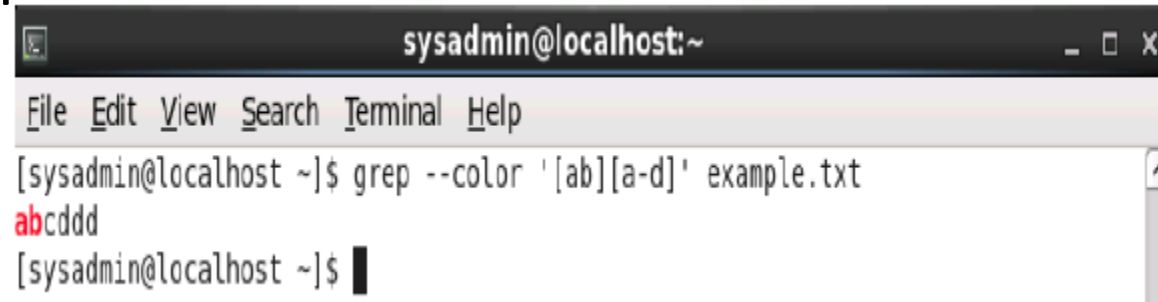
- Exemple 2 :on cherche les termes correspondant à 4 caractères le premier étant « a » et le 4ème « c »



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ grep --color 'a..c' example.txt  
[sysadmin@localhost ~]$
```

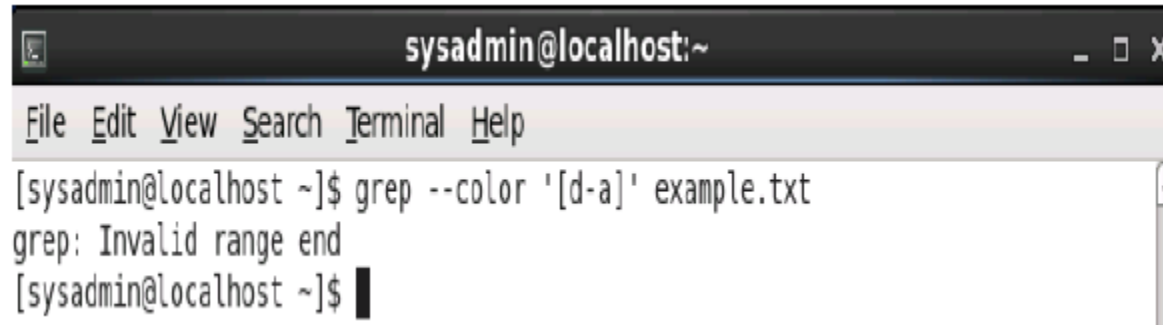
Les expressions régulières de base (bre):les []

- **Exemple1**: dans le fichier **example.txt** on cherche les termes correspondant à 2 caractères (le premier a ou b et le deuxième a.b.c ou d)



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ grep --color '[ab][a-d]' example.txt  
abdddd  
[sysadmin@localhost ~]$
```

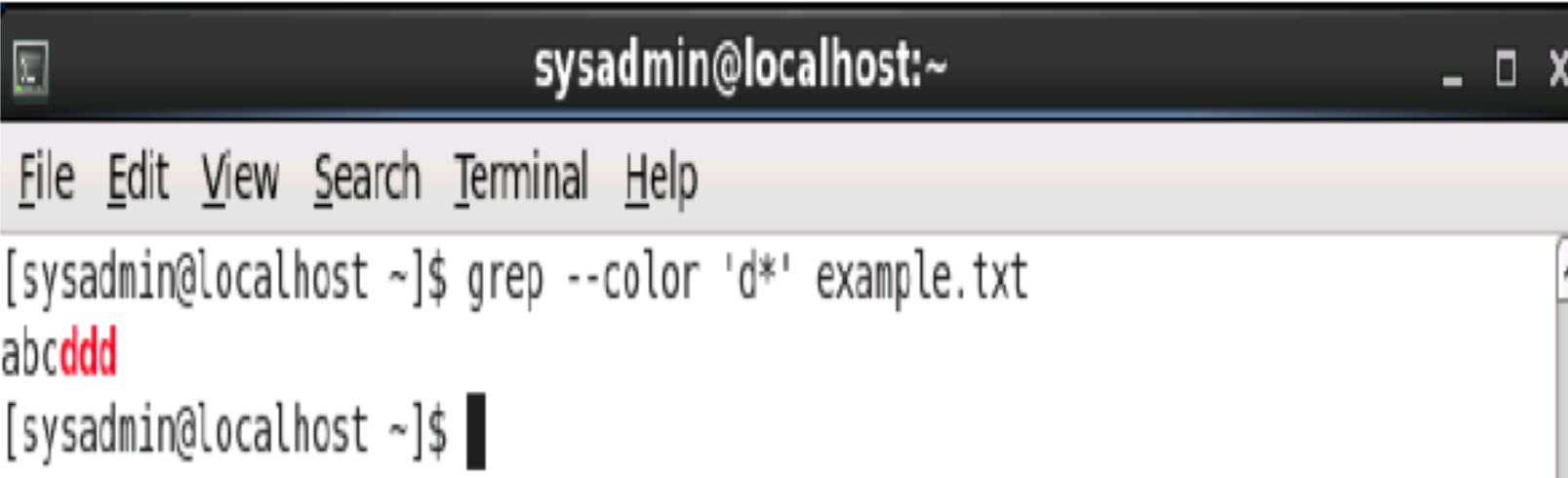
- **Exemple 2** : ici *grep* donne erreur car l'ordre des caractères n'est pas correcte , l'ordre est spécifié par le standard ascii (man ascii)



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ grep --color '[d-a]' example.txt  
grep: Invalid range end  
[sysadmin@localhost ~]$
```

Les expressions régulières de base (bre):le caractère *

- **Exemple** : on cherche les terme contenant 0 caractère d ou plus

A terminal window titled 'sysadmin@localhost:~' with a menu bar containing 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal shows the command '[sysadmin@localhost ~]\$ grep --color 'd*' example.txt' and its output 'abcddd', where the 'ddd' part is highlighted in red. The prompt '[sysadmin@localhost ~]\$' is followed by a cursor.

```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ grep --color 'd*' example.txt  
abcddd  
[sysadmin@localhost ~]$
```

Les expressions régulières de base (bre):les caractères ^ et \$

- Exemple1 : le caractère

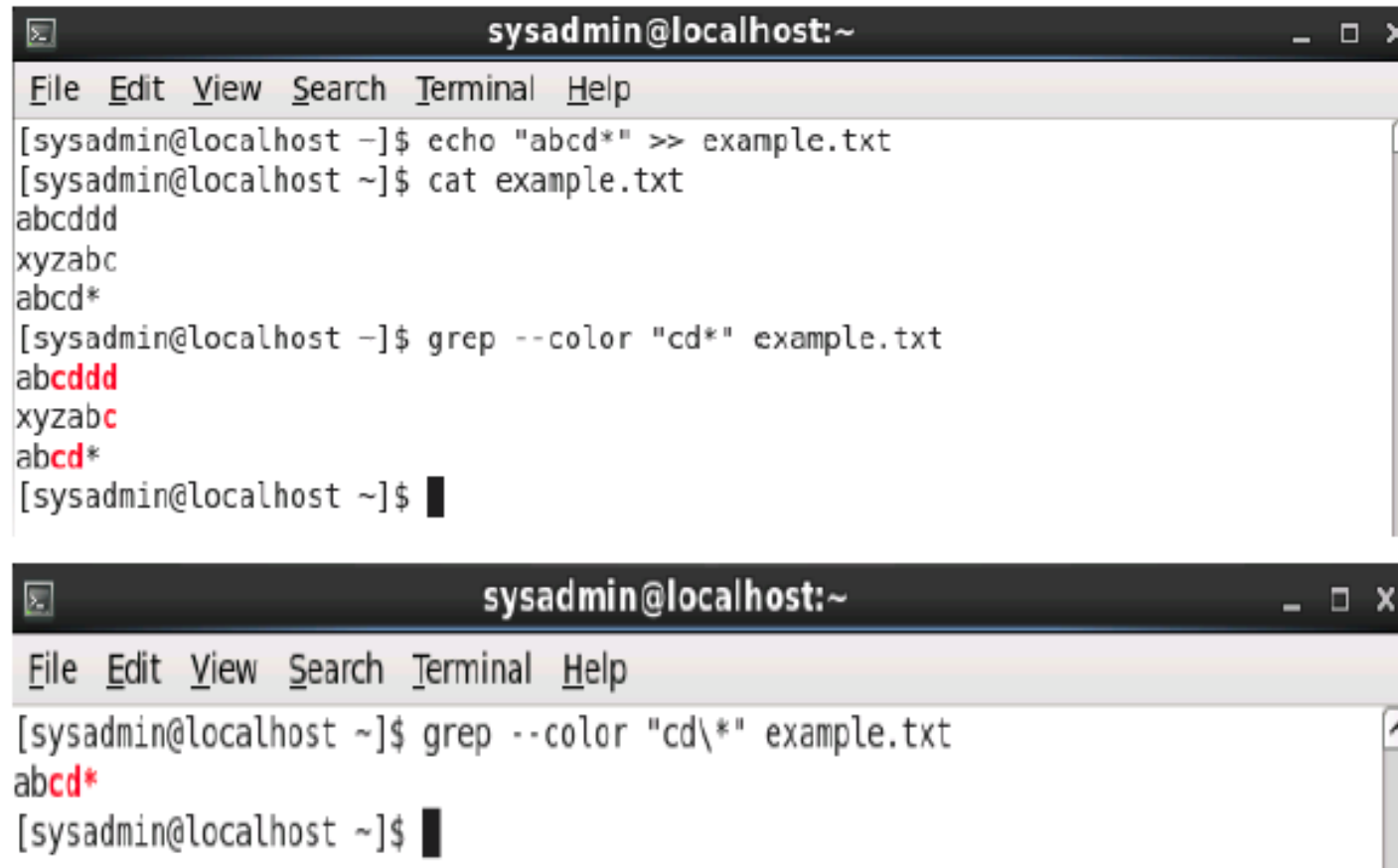
```
sysadmin@localhost:~$ echo "xyzabc" >> example.txt
sysadmin@localhost:~$ cat example.txt
abcddd
xyzabc
sysadmin@localhost:~$ grep --color "a" example.txt
abcddd
xyzabc
sysadmin@localhost:~$ grep --color "^a" example.txt
abcddd
sysadmin@localhost:~$
```

- Exemple2:le caractère \$

```
sysadmin@localhost:~$ grep "c$" example.txt
xyzabc
sysadmin@localhost:~$
```

Les expressions régulières de base (bre):le caractère \

- Le caractère \ permet d'annuler l'interprétation d'un caractère spécial dans une expression régulière
- exemple



```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ echo "abcd*" >> example.txt  
[sysadmin@localhost ~]$ cat example.txt  
abcddd  
xyzabc  
abcd*  
[sysadmin@localhost ~]$ grep --color "cd*" example.txt  
abcddd  
xyzabc  
abcd*  
[sysadmin@localhost ~]$
```

```
sysadmin@localhost:~  
File Edit View Search Terminal Help  
[sysadmin@localhost ~]$ grep --color "cd\" example.txt  
abcd*  
[sysadmin@localhost ~]$
```

Les expressions régulières étendues

- Pour utiliser les expressions régulières avec une commande il faut spécifier une option supplémentaire , pour la commande **grep** il faut utiliser l'option **-E** ou la commande **egrep** (**egrep** veut dire la même chose que **grep -E**) le tableau suivant donne des exemples d'expressions régulières étendues

expression	signification
?	Le caractère précédent figure 0 ou une fois
+	Le caractère précédent repete 1 ou plusieurs fois
	Agit comme un ou logique

Les expressions régulières étendues (exemples)

commande	signification	Exemple de résultat
<code>grep -E 'colou?r' 2.txt</code>	Les termes contenant « colo » suivi par 0 ou 1 u	color colour
<code>grep -E 'd+' 2.txt</code>	Les termes contenant 1 d ou plus	d dd ddd
<code>grep -E 'gray grey' 2.txt</code>	Les termes contenant grey ou gray	grey gray

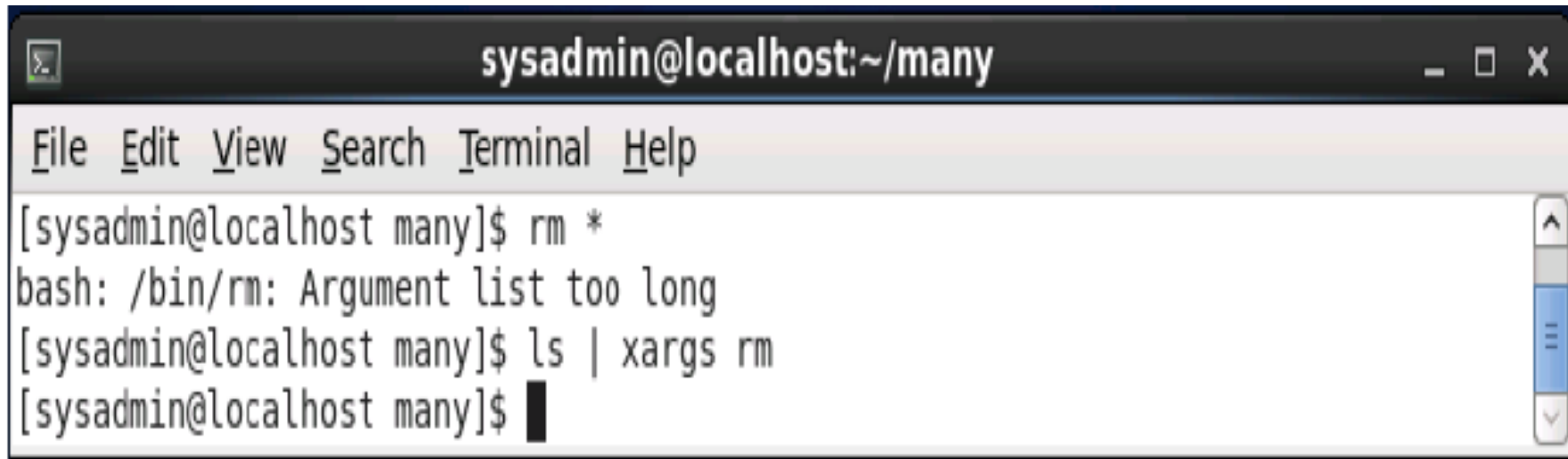
La commande xargs

- La commande **xargs** prend comme argument une commande puis lit les paramètres de la commande à partir de l'entrée standard(ou autre flux), puis exécute la commande passée comme argument avec les paramètres fournis
- Exemple

```
~$ xargs ls
-|
/etc/ppp
total 76
-rw----- 1 root root 80 juil. 22 2014 chap-secrets
-rwxr-xr-x 1 root root 1754 janv. 22 2013 ip-down
drwxr-xr-x 2 root root 4096 déc. 26 11:01 ip-down.d
-rwxr-xr-x 1 root root 1892 janv. 22 2013 ip-up
```

La commande xargs

- **Exemple2:**ici on passe comme argument à **xargs** la commande **rm** puis lit les paramètres à partir de la sortie de la commande **ls**

A terminal window titled 'sysadmin@localhost:~/many' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command 'rm *' failing with the error 'bash: /bin/rm: Argument list too long'. Then, the command 'ls | xargs rm' is entered, and the prompt returns without further output.

```
sysadmin@localhost:~/many
File Edit View Search Terminal Help
[sysadmin@localhost many]$ rm *
bash: /bin/rm: Argument list too long
[sysadmin@localhost many]$ ls | xargs rm
[sysadmin@localhost many]$
```

Autres commandes de manipulation de fichier

- `uniq [options] fichier`
- La commande **uniq** permet de supprimer les lignes redondantes consécutives dans un fichier, l'option `-c` : précède chaque ligne du résultat du nombre d'occurrences de cette ligne dans le fichier original
- **exemple**

```
$ cat exemple.txt  
Ceci est un test.  
Ceci est un test.  
unix  
TEST  
TEST
```

```
$ uniq exemple.txt  
Ceci est un test.  
unix  
TEST
```

```
$ uniq -c exemple.txt  
2 Ceci est un test.  
1 unix  
2 TEST
```

Autres commandes de manipulation de fichier

- `diff [options] fichier1 fichier2`
- La commande `diff` réalise la comparaison ligne à ligne du fichier texte « fichier2 » par rapport au fichier texte « fichier1 ».
- Exemple:

```
$ cat fichier1
1 Blabla bla bla.
2 Deux fotes d'ortographe
ici.
3 Encore du blabla bla
bla.
```

```
$ cat fichier2
1 Blabla bla bla.
2 Deux fautes
d'ortographe ici.
3 Encore du blabla bla
bla.
```

```
$ diff fichier1 fichier2
2c2
< Deux fotes
d'ortographe ici.
---
> Deux fautes
d'orthographe ici.
Interprétation de
«2c2» : la ligne 2 de
«fichier2» est changée
par rapport à la ligne 2
de «fichier1 »
```

Autres commandes de manipulation de fichier

- La commande **split** permet de couper un fichier en morceau (en plusieurs fichiers), en tapant : ***split -n monfichier fichier***
 - Vous allez créer les fichiers fichieraa, fichierab, fichierac, ... Qui contiendront tous n lignes. Le premier fichieraa contient les n premières lignes, ainsi de suite.
- La commande **tee** permet de lire à partir de l'entrée standard et écrire dans la sortie standard ou dans un fichier elle surtout est utilisée avec les pipes

```
[sysadmin@localhost ~]$ ls -l /etc/ppp | tee | head -5
total 48
-rw-----. 1 root root  78 10 juin  2014 chap-secrets
-rw-----. 1 root root 349 10 juin  2014 eaptls-client
-rw-----. 1 root root 405 10 juin  2014 eaptls-server
-rwxr-xr-x. 1 root root 386  3 mai   2017 ip-down
[sysadmin@localhost ~]$
```