



Cours Linux^{v.2}

« *Les commandes avancées* »

Frédéric BURLET

François SCHOUBBEN

Louis SWINNEN

{fred,fsc,louis}@namurlug.org

12 mars 2003

Copyright information

Ces slides sont fournis sous licence **GNU Free Documentation License**
These slides are provided under the **Free Documentation License**

Copyright (C) 2003 Namur LUG

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts and with no Back-Cover Texts

A copy of the license is included in the file "[fdl.txt](#)".

For more information about *free licenses*, see the GNU project website.

Source of this document can be obtained by mail to the authors.

Powered by OpenOffice 1.0.1

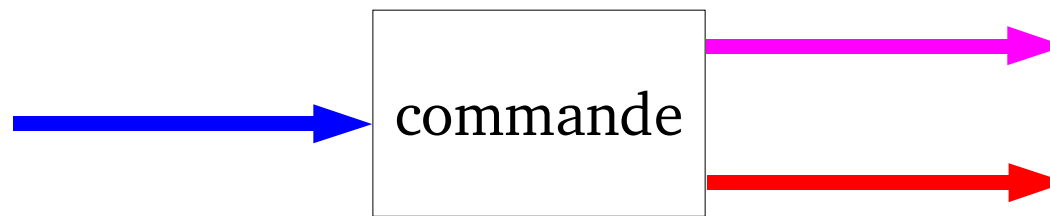
Commandes avancées (1)

Redirection (1)

- Les flux standards

- **stdin**: l'entrée standard ← → clavier
- **stdout**: la sortie standard ← → écran
- **stderr**: le standard d'erreur ← → écran

par défaut



Toutes les commandes n'ont pas forcément d'entrée et/ou de sortie standard/erreur

Commandes avancées (1)

Redirection (2)

- Redirection de la **sortie standard**

- Dans un fichier

- En utilisant les symboles '>' et '>>'

- Commande:

`commande > fichier`

`commande >> fichier`

- Exemples:

`$ ls > out`

`$ ls -l >> out`

`$ ls /root > out`

ce qui était imprimé
à l'écran est redirigé

← crée le fichier ou
écrase s'il existe

← ajoute à la fin du fichier
s'il existe et le crée sinon

**Les erreurs ne sont
pas redirigées !**

Commandes avancées (1)

Redirection (3)

- Redirection du **standard d'erreur**

- Dans un fichier
- En utilisant le symbole '2>'
- Commande:

`commande 2> fichier`

`commande &> fichier`

- Exemples:

```
$ ls /root 2> out
```

```
$ ls /* &> out
```

les messages d'erreurs
sont redirigés

← Redirige le standard
d'erreur

← sortie standard+erreur

Commandes avancées (1)

Redirection (4)

- Redirection de l'entrée standard

- A partir d'un fichier
- En utilisant le symbole '<'
- Commande:
commande < fichier

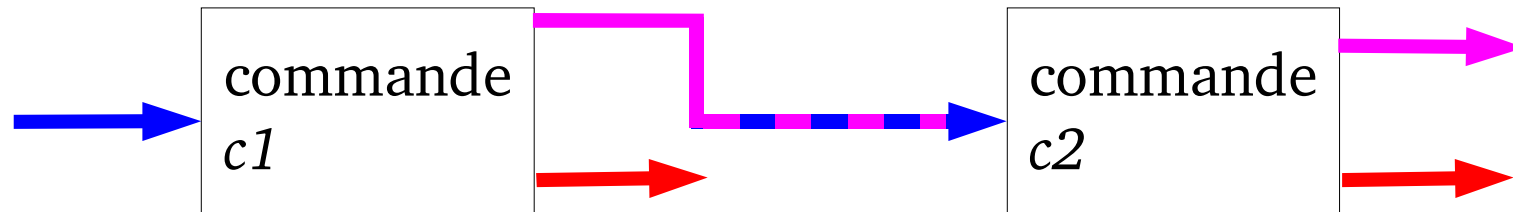
- Exemple:

```
$ tr a-mn-z n-za-m < /etc/inittab
```

*ce qui était demandé
au clavier est redirigé*

Commandes avancées (2)

Le pipe (1)



Idée: Exécuter deux commandes *c1* et *c2* de sorte que:

- L'**entrée standard** de *c2* corresponde à la **sortie standard** de *c1*
- Le résultat de *c1* sert de donnée d'entrée pour la commande *c2*

Fortement utilisé dans les systèmes UNIX !

Commandes avancées (2)

Le pipe (2)

– Mise en oeuvre

- Permet de **lier deux commandes**
- Utilisation du symbole '|'

- Principe:

`c1 | c2 | ...`

La sortie de *c1* constitue les données d'entrée de *c2*, ...

- Exemple:

```
$ ls -l /home/rouge/students | grep teacher
```

```
$ cat /usr/bin/sambamontage | grep "leda"
```


Commandes avancées (3)

Principes (1)

- Système multitache
 - Linux est un système capable de faire tourner plusieurs programmes en même temps
 - Plusieurs programmes serveurs
 - Plusieurs applications
 - Ce système est robuste
 - Une tâche qui boucle ne remet pas en cause l'intégrité du système
 - Possibilité de demander/d'obliger l'arrêt d'un programme

Commandes avancées (3)

Principes (2)

- Système multi-utilisateur
 - La première opération que l'utilisateur doit faire est s'authentifier
 - Utilisateur et groupe
 - Superutilisateur ou root
 - droits différents
 - Gestion de plusieurs utilisateurs pouvant être connectés « en même temps » sur la machine.
 - Sur plusieurs console
 - A distance

Commandes avancées (4)

Permission (1)

- Droits d'accès (fichier et répertoire)
 - 3 types de droits
 - autorisation de lire
 - autorisation d'écrire
 - autorisation d'exécuter/de traverser
 - 3 groupes différents:
 - le propriétaire
 - Celui qui a créé le fichier/répertoire
 - le groupe auquel le propriétaire appartient
 - Le groupe principal du propriétaire
 - les autres
 - Le reste du monde

Commandes avancées (4)

Permission (2)

– Cas des répertoires:

- Le droit en lecture indique que l'on autorise (ou pas) la *lecture du contenu du répertoire*
- Le droit en écriture indique que l'on autorise (ou pas) l'*écriture dans le répertoire*
- Le droit en exécution indique que l'on autorise (ou pas) à *traverser le répertoire*
- Exemple:

drwxr-xr-x root bin games

The diagram shows the permission string 'drwxr-xr-x' with three colored arrows pointing to its components: a red arrow from the first 'r' to 'droits pour le propriétaire (root)', a green arrow from the second 'r' to 'droits pour le groupe (bin)', and a black arrow from the third 'r' to 'droits pour les autres'.

Commandes avancées (4)

Permission (3)

– Cas des fichiers

- Le droit en lecture indique que l'on autorise (ou pas) la *lecture du contenu du fichier*
- Le droit en écriture indique que l'on autorise (ou pas) l'*écriture dans le fichier*
- Le droit en exécution indique que l'on autorise (ou pas) l'*exécution du fichier*
- Exemple:

`-rwxr-xr-x` `teacher` `staff` `path.sh`

droits pour les autres

droits pour le groupe (staff)

droits pour le propriétaire (teacher)

Commandes avancées (5)

- Modification des permissions

- chmod

- Permet de **changer les permissions** d'un fichier ou d'un répertoire

- Commande:

```
chmod [-R] [augo±rwx] fichier|répertoire
```

- Exemple:

r: lecture
w: écriture
x: exécution

```
$ chmod ugo+x doc/francois  
$ ls -l  
$ chmod -R go-rx doc/louis  
$ ls -l  
$ chmod u=rwx,g=rx,o=- doc
```

u: user (propriétaire)
g: groupe
o: other (autres)
a: ugo

Commandes avancées (6)

– chmod (autre syntaxe)

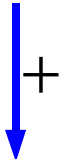
- Permet de **fixer** une permission

- Commande

```
chmod [-R] perm fichier...
```

- 'perm':

	user	group	other
read	400	40	4
write	200	20	2
exec	100	10	1



- Exemple:

```
$ chmod 774 doc
```

Commandes avancées (7)

Administration (1)

- Groupes et utilisateurs
 - Le système supporte plusieurs utilisateurs
 - Chaque utilisateur est associé à un groupe principal et une multitude de groupes secondaires.
 - Des droits et permissions peuvent être accordés sur chaque groupe
 - Seul l'**administrateur** peut créer et modifier des groupes

Commandes avancées (7)

Administration (2)

– useradd

- Permet d'**ajouter un utilisateur** dans le système
- Cette commande crée le répertoire personnel de l'utilisateur
- Commande:
`useradd [-g gid] username`
- **Commande réservée à l'administrateur**
- L'utilisateur ne sera actif que **lorsque son mot de passe aura été défini**
- Exemple:
`$ useradd -g users lswinnen`

Commandes avancées (7)

Administration (3)

– passwd

- Permet **de changer le mot de passe** d'un utilisateur
- Seul l'administrateur peut changer le mot de passe d'un autre utilisateur

- Commande:

```
passwd [-l|-u] [username]
```

- Exemple:

```
$ passwd lswinnen
```

Commandes avancées (7)

Administration (4)

– userdel

- Permet de **supprimer un utilisateur**
- Réservé à l'**administrateur**
- Seul un utilisateur non connecté peut être supprimé.
- Par défaut, le répertoire personnel n'est pas supprimé
- Commande:
`userdel [-r] username`
- Exemple:
`$ userdel lswinnen`

Commandes avancées (7)

Administration (5)

– groupadd

- Permet d'**ajouter un groupe** au système

- Réservé à l'administrateur

- Commande:

```
groupadd [-g gid] groupname
```

- Exemple:

```
$ groupadd web
```

Commandes avancées (7)

Administration (6)

– groupdel

- Permet **de supprimer un groupe**
- Il est interdit de supprimer le groupe principal d'un utilisateur
- Commande:
`groupdel groupname`
- Exemple:
`$ groupdel web`

Commandes avancées (7)

Administration (7)

- Fichier des utilisateurs
 - Le fichier `/etc/passwd` décrit les utilisateurs du système
 - Chaque utilisateur est renseigné avec son groupe et d'autres informations (shell, ...)
 - Pour des raisons de sécurité, le fichier ne contient pas le mot de passe de l'utilisateur
 - Ce fichiers doit être accessible en lecture pour tout le monde (contraintes techniques)

Commandes avancées (7)

Administration (8)

- Fichier des groupes
 - Le fichier `/etc/group` **décrit tous les groupes** présent dans le système
 - Chaque groupe contient les utilisateurs qui en font partie
 - Un utilisateur peut donc se retrouver dans plusieurs groupes différents.

Commandes avancées (7)

Administration (9)

- Changement de propriétaire/groupe
 - chown
 - Permet de changer le propriétaire (et le groupe) d'un fichier
 - Seul l'administrateur peut changer le propriétaire d'un fichier
 - Commande:
`chown user[.group] fichier...`
 - Exemple:
`$ chown lswinnen.web fichier`

Commandes avancées (7)

Administration (10)

– chgrp

- Permet de **changer groupe** d'un fichier/répertoire
- Un utilisateur peut changer le groupe d'un fichier/répertoire s'il fait partie du groupe destination

- Commande

```
chgrp groupe fichier...
```

- Exemple:

```
$ chgrp web tmp
```

Commandes avancées (8)

Processus (1)

- Gestion des processus
 - Un processus est **un programme en cours d'exécution**
 - Lorsqu'un programme est lancé, le système lui associe un numéro: le *PID* (*process ID*)
 - Il est possible d'envoyer des signaux à des processus (SIGUSR, SIGKILL, SIGINT)
 - Le signal SIGKILL permet de terminer l'exécution d'un processus. Ex: s'il boucle !

Commandes avancées (8)

Processus (2)

- Lancement de tâche en arrière plan
 - Une application peut s'exécuter en tâche de fond

- Commande:

commande &

- Exemple:

```
$ mozilla &
```

```
$ emacs
```

```
[2] 24567
```

↓ PID de l'application lancée

↓ Numéro de la tâche associé à ce terminal

Commandes avancées (8)

Processus (3)

- CTRL-Z (^ Z)
 - permet de **suspendre l'exécution d'un programme** lancé à partir d'un terminal
 - Faire ^ Z dans le terminal
 - Exemple:
^Z

Commandes avancées (8)

Processus (4)

– jobs

- Permet **de connaître les processus lancé dans ce terminal**
- A faire dans le terminal
- Exemple:

```
$ jobs
```

Commandes avancées (8)

Processus (5)

– bg (background)

- Permet **de relancer un processus suspendu en tâche de fond**
- A faire dans le terminal
- Exemple:

bg (dans le terminal)

– fg (foreground)

- Permet **de relancer un processus suspendu ou en tâche de fond en premier plan**
- Exemple:

fg %1 (dans le terminal)

Commandes avancées (8)

Processus (6)

- `ps` (process status)
 - Permet d'**obtenir la liste des processus**
 - Commande:
`ps [-e|-f|x|-u name]`
 - Exemples:

```
$ ps -e -f x
```

```
$ ps -f -u root
```

```
$ ps -e -f x | grep "gdm"
```

```
$ ps -f -u visitr_stX | grep "bash"
```

Commandes avancées (8)

Processus (7)

- **kill**

- Permet d'envoyer un signal donné à un processus

- A chaque signal est associé un numéro

- Commande:

```
kill [-s signal] pid
```

- Exemple:

```
$ kill -s kill <pid_emacs>
```

signaux courant:

term: arrêt du processus (15)

kill: suppression du proc. (9)

hup: action spécifique (1)

Commandes avancées (8)

Processus (8)

- top
 - Affiche l'état du système et les processus en cours
 - Commande:
top
 - Exemple:
\$ top

Utilisation de top:

? - aide

q – quitter

u – pour *un* utilisateur

k – envoi d'un signal

P – tri sur le CPU

M – tri sur la mémoire

Commandes avancées (8)

Processus (9)

– pstree

- Affiche un **arbre des processus en cours**
- Montre les processus parents/enfants

- Commande:

```
pstree
```

- Exemple:

```
$ pstree
```

Commandes avancées (8)

Processus (10)

- Panique !
 - CTRL-C (^ C)
 - Permet de **terminer un processus** (= kill-s TERM)
 - CTRL-ALT-BACKSPACE
 - Permet de **redémarrer l'interface graphique** quand tout semble "planté"
 - Supprime les processus lancés