

UE : Système d'exploitation

Licence 3 SFA Informatique

- Interpréteur de commandes
 - Interactif
 - Non-Interactif
 - Login
 - Non-Login
- Fichiers de démarrage
- Permissions
 - Principe
 - Permissions spéciales
 - Des commandes
 - Manipulations avec la commande chmod
 - La représentation en octal
 - Substitution d'utilisateur
 - Substitution de super-utilisateur
 - Et encore...

Interpréteur de commandes interactif

Shell interactif

- Les commandes sont exécutées via l'interaction d'un utilisateur au clavier. Le shell peut prendre en compte des entrées utilisateur.
- C'est le cas des commandes exécutées directement dans un terminal
- Exemple :

```
damien@dgn:~$ env | grep TEST
TESTBASHRC=1
TESTPROFILE=2
damien@dgn:~$
```

Interpréteur de commandes non-interactif

Shell non-interactif

- Le shell ne peut savoir si des entrées utilisateur sont possibles, ni si la sortie sera affichée à celui-ci.
- C'est le cas des commandes exécutées depuis un autre processus
- Exemple :

```
damien@dgn:~$ ssh damien@localhost "env | grep TEST"
damien@localhost's password:
damien@dgn:~$ ssh damien@localhost "ls"
damien@localhost's password:
apache-jmeter-5.3
apps
```

Login shell

Login shell

- Un shell dans lequel vous vous connectez avant d'arriver dans l'interpréteur
- Exemple : Connexion à une session **ssh**

```
testuser@dgn:/home/damien$ ssh damien@localhost
damien@localhost's password:
Last login: Tue Nov 17 14:56:24 2020 from 127.0.0.1
damien@dgn:~$ env | grep TEST
TESTBASHRC=1
TESTPROFILE=2
damien@dgn:~$ logout
Connection to localhost closed.
testuser@dgn:/home/damien$
```

Non-login shell

Non-login shell

- Un shell ouvert sans qu'une **vraie** authentification ait été réalisée
- Exemple : Se connecter “en tant que” via la commande **su**

```
root@dgn:/# su damien
damien@dgn:/$ env | grep TEST
TESTBASHRC=1
damien@dgn:/$ exit
root@dgn:/#
```

Les fichiers de démarrage

- **/etc/bashrc** ou **/etc/bash.bashrc**
 - Mais aussi : **\$HOME/.bashrc**
 - “Sourcé” depuis les **shell interactifs** et les **login shell interactifs**
 - Spécifique à l’interpréteur **BASH**
- **/etc/profile**
 - Mais aussi : **\$HOME/.profile**
 - “Sourcé” depuis les **login shell interactifs**

Les fichiers de démarrage

Les rôles des fichiers **bashrc** et **profile** :

- Définir toutes sortes de choses qui seront systématiquement utilisées dans un shell
 - Par exemple :
 - Définir des variables (exemple: **\$PATH**)
 - Définir des alias
 - Définir des fonctions
 - ...

Les fichiers de démarrage

Si présent, le fichier **.profile** qui est dans le répertoire “home” d’un utilisateur sera sourcé **APRÈS** le fichier **/etc/profile** qui est global au système.

Le même principe s’applique avec le fichier **/etc/bashrc**.

Ce mécanisme permet aux utilisateurs d’apporter des configurations spécifiques à leur session en supplément de celles qui sont globales au système.

Attention, les noms de ces fichiers varient parfois d’un système à l’autre.

Les fichiers de démarrage : Un exemple

Imaginons qu'on définisse des **variables d'environnement** comme ceci :

- Le fichier **/etc/bash.bashrc** contient : `export TESTBASHRC=1`
- Le fichier **/etc/profile** contient : `export TESTPROFILE=1`
- Le fichier **/home/damien/.profile** contient : `export TESTPROFILE=2`

Les fichiers de démarrage : Un exemple

En tant que utilisateur **damien**, on observe que les fichiers **/etc/bash.bashrc** et **/etc/profile** puis **.profile** ont été sourcés.

Ensuite, on lance un **shell interactif de type login** (option -i) en tant que utilisateur **testuser** grâce à la commande **sudo**.

On constate que le fichier **/etc/bash.bashrc** n'a pas été sourcé, car l'interpréteur par défaut n'est pas **/etc/bash** mais **/etc/sh**.

On constate aussi que la variable **TESTPROFILE** est égale à **1** au lieu de **2** car l'utilisateur courant n'est pas **damien** mais **testuser** : le fichier **/home/damien/.profile** n'a donc pas été sourcé.

```
damien@dgn:~$ whoami
damien
damien@dgn:~$ env | grep TEST
TESTBASHRC=1
TESTPROFILE=2
```

```
damien@dgn:~$ sudo -i -u testuser
$ whoami
testuser
$ env | grep TEST
TESTPROFILE=1
$ echo $SHELL
/bin/sh
$
$ /bin/bash
testuser@dgn:~$ env | grep TEST
TESTBASHRC=1
TESTPROFILE=1
```

Les fichiers de démarrage

La modification des fichiers de démarrage peut nécessiter de relancer son shell ou de redémarrer le système pour être effective.

Mais il est possible de les “sourcer” manuellement dans le shell actuel avec la commande **source** :

```
dg@DESKTOP-B39B320:~$ echo $MA_VARIABLE  
  
dg@DESKTOP-B39B320:~$ source .profile  
dg@DESKTOP-B39B320:~$ echo $MA_VARIABLE  
1
```

Les permissions : Le principe

Sous Linux, les **utilisateurs** appartiennent à un **groupe**, et il existe un super-utilisateur appelé **root**.

Et chaque fichier possède :

- Un **propriétaire** (utilisateur)
- Un **groupe**
- Un **ensemble de permissions**

Les permissions : Le principe

Les permissions sur un fichier sont découpées en 3 parties :

- Les permissions que possède **le propriétaire**
- Les permissions que possède **le groupe**
- Les permissions que possèdent **les autres**

Les permissions : Le principe

Et chaque partie est découpée en 3 autorisations :

- L'autorisation de **lire un fichier**, ou de **lister le contenu d'un dossier**
- L'autorisation d'**écrire dans un fichier ou dans un dossier**
- L'autorisation d'**exécuter un fichier**, ou de **se déplacer dans un dossier**

Les permissions : Le principe

Ici un exemple avec la commande `ls -l`

```
-rw-r--r-- 1 dg users 4 Nov 16 21:08 myfile
```

- La **première partie** (-rw-r--r--) représente **l'ensemble des permissions**.
- La colonne avec écrit "**dg**" indique le **propriétaire** du fichier.
- La colonne avec écrit "**users**" indique le **groupe** auquel appartient le fichier.

Les permissions : Le principe

Que signifie cet exemple concrètement ?

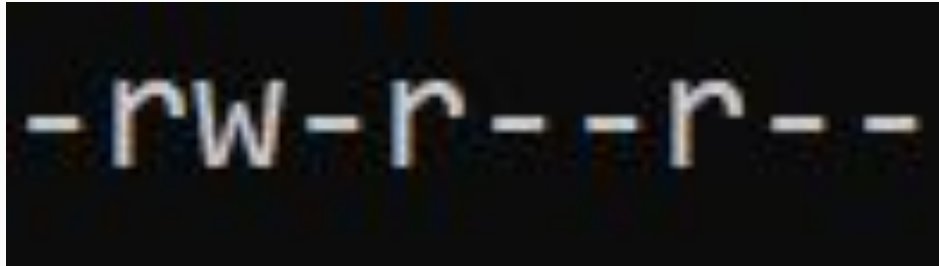
```
-rw-r--r-- 1 dg users 4 Nov 16 21:08 myfile
```

Et bien déjà on peut dire que :

- **Le propriétaire** du fichier est l'utilisateur **dg**
- **Le groupe** auquel appartient le fichier est le groupe **users**

Les permissions : Le principe

Et ensuite ?



-rw-r--r--

Les permissions : Le principe

D'abord la première “case” donne une indication sur le type de fichier.

```
lrwxrwxrwx 1 root root    9 Aug  3  2019 lock -> /run/lock
drwxr-xr-x 1 root root 512 Jun 13 01:52 log
-rw-r--r-- 1 root root   0 Nov 16 21:16 lol
```

Ici on voit que :

- (-) Le fichier nommé “**lol**” est un simple **fichier**
- (d) Le fichier nommé “**log**” est un **dossier**
- (l) Le fichier nommé “**lock**” est un **lien symbolique**

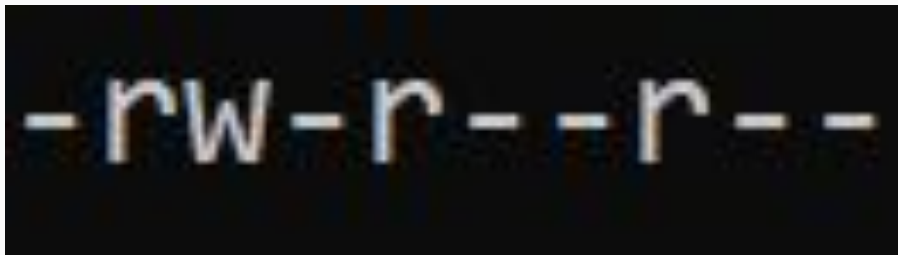
Les permissions : Le principe

Ensuite :

- On décompose en 3 parties : **Propriétaire**, puis **groupe**, puis **autres**
- Chaque partie décomposée en 3 autorisations : **Lire**, puis **écrire**, puis **exécuter**

Ce qui nous donne :

- **Propriétaire** : rw- (lire, écrire)
- **Groupe** : r-- (lire)
- **Autres** : r-- (lire)



-rw-r--r--

Les permissions : Le principe

Un autre exemple

```
drwxr-xr-x 1 root root 512 Apr 26 2020 home
-rwxr-xr-x 1 root root 631968 Jan 1 1970 init
drwxr-xr-x 1 root root 512 Apr 26 2020 lib
drwxr-xr-x 1 root root 512 Aug 3 2019 lib64
drwx----- 1 root root 512 Apr 10 2019 lost+found
drwxr-xr-x 1 root root 512 Aug 3 2019 media
drwxr-xr-x 1 root root 512 Jun 5 14:03 mnt
drwxr-xr-x 1 root root 512 Aug 3 2019 opt
dr-xr-xr-x 9 root root 0 Nov 16 20:59 proc
drwx----- 1 root root 512 Jun 22 10:32 root
drwxr-xr-x 1 root root 512 Nov 16 21:09 run
drwxr-xr-x 1 root root 512 May 27 11:04 sbin
drwxr-xr-x 1 root root 512 Aug 3 2019 srv
dr-xr-xr-x 12 root root 0 Nov 16 20:59 sys
drwxrwxrwt 1 root root 512 Nov 13 23:03 tmp
```

Des permissions un peu spéciales

Il existe d'autres types de permissions qu'il est bon de connaître :

- Sticky bit

```
damien@dgn:~$ ls -ld /tmp
drwxrwxrwt 18 root root 4096 nov. 21 11:25 /tmp
```

- SUID

```
damien@dgn:~$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 68208 mai 28 08:37 /usr/bin/passwd
```

- SGID

```
damien@dgn:~$ ls -l /usr/bin/crontab
-rwxr-sr-x 1 root crontab 43720 févr. 13 2020 /usr/bin/crontab
```

De nombreuses commandes

De nombreuses commandes permettent de manipuler les utilisateurs, groupes et permissions :

- Changement de permissions, propriétaire et groupe :
 - `chmod`, `chown`, `chgrp`...
- Manipulation d'utilisateurs et de groupes :
 - `adduser`, `deluser`, `addgroup`, `delgroup`...
- Autres :
 - `umask`, `stat`, `passwd`, `usermod`, `sudo`, `su`...

Manipulations avec la commande chmod

La commande **chmod** permet d'**ajouter**, de **supprimer**, ou de **définir** les permissions appliquées sur un fichier.

En utilisant la forme vue précédemment avec la commande **ls -l**, on pourrait tout simplement définir des droits comme ceci :

```
damien@dgn:~$ chmod u=rwx,g=rw,o= file
damien@dgn:~$ ls -l file
-rwxrw--- 1 damien damien 0 nov. 21 17:56 file
```

Ici **u=** définit les droits du propriétaire, **g=** du groupe et **o=** des autres.

Manipulations avec la commande chmod

Au lieu de **définir** des permissions, on peut en **ajouter** ou en **supprimer** en remplaçant le signe “égal” = par un signe “plus” + ou par un “signe” - dans la commande précédente :

```
damien@dgn:~$ chmod u-x,o+r file
damien@dgn:~$ ls -l file
-rw-rw-r-- 1 damien damien 0 nov. 21 17:56 file
```

Ici on **supprime** les droits d'**exécution** au propriétaire et on **ajoute** les droits de **lecture** aux autres.

Représentation en octal

Les permissions peuvent s'écrire avec des **chiffres** sous forme **octale**. C'est à dire en "base 8".

Par exemple : Définir les droits à **764** revient à définir les droits suivants : u=**rw**x,g=**rw**,o=**r**

Octal	Binary	File Mode
0	000	- - -
1	001	- - x
2	010	- w -
3	011	- w x
4	100	r - -
5	101	r - x
6	110	r w -
7	111	r w x

```
damien@dgn:~$ chmod 764 file
damien@dgn:~$ ls -l file
-rwxrw-r-- 1 damien damien 0 nov. 21 17:56 file
```

Substitution d'utilisateur

Substitution d'utilisateur :

- Il est possible d'exécuter une commande en tant qu'un autre utilisateur avec la commande **sudo** (Substitute User **DO**).
- Il est possible d'ouvrir un shell en tant qu'un autre utilisateur avec la commande **su** (Substitute User).

```
dg@DESKTOP-B39B320:~$ sudo su root
root@DESKTOP-B39B320:/home/dg#
root@DESKTOP-B39B320:/home/dg# sudo -u dg whoami
dg
root@DESKTOP-B39B320:/home/dg# exit
dg@DESKTOP-B39B320:~$ sudo whoami
root
```

Substitution de super-utilisateur

Les commandes **sudo** et **su** permettent aussi d'exécuter des commandes en tant que **root**.

Néanmoins il est nécessaire que votre utilisateur figure dans la liste des **sudoers** pour permettre d'invoquer les droits super-utilisateur depuis la commande **sudo**.

```
dg@DESKTOP-B39B320:~$ sudo cat /etc/sudoers | grep sudo
# This file MUST be edited with the 'visudo' command as root.
# Please consider adding local content in /etc/sudoers.d/ instead of
# See the man page for details on how to write a sudoers file.
# Allow members of group sudo to execute any command
%sudo  ALL=(ALL:ALL) ALL
# See sudoers(5) for more information on "#include" directives:
#includedir /etc/sudoers.d
dg@DESKTOP-B39B320:~$ cat /etc/group | grep sudo
sudo:x:27:dg
```

Les permissions c'est encore plus complet

D'autres choses qui existent que vous pouvez creuser :

- Les ACL : Access Control Lists
 - Une autre forme de manipulation des permissions, plus avancée
- Le umask
 - Définir un masque de permissions qui **réduisent** les permissions par défaut qui seront appliquées à des fichiers nouvellement créés

```
damien@dgn:~$ umask 077 ; touch file && ls -l file
-rw----- 1 damien damien 0 nov. 21 11:58 file
damien@dgn:~$
damien@dgn:~$ rm file
damien@dgn:~$
damien@dgn:~$ umask 022 ; touch file && ls -l file
-rw-r--r-- 1 damien damien 0 nov. 21 11:58 file
```

- Wikipédia
- https://www.stefaanlippens.net/bashrc_and_others/
- https://wiki.archlinux.org/index.php/File_permissions_and_attributes
- (s) <https://en.wikipedia.org/wiki/Setuid>
- (t) https://en.wikipedia.org/wiki/Sticky_bit
- <https://www.thegeekdiary.com/what-is-suid-sgid-and-sticky-bit/>
- <https://doc.ubuntu-fr.org/sudoers>
-