UPMC
PARIS UNIVERSITAS

Préparation LPI

Exam 101

104.5 Permissions et appartenance des fichiers

104.5 – Domaines de connaissances

- Poids : 3
- Gérer les permissions de fichiers (réguliers et spéciaux) et répertoires
- Utiliser les modes d'accès suid, sgid et sticky bit pour gérer la sécurité
- Comment changer un masque de création de fichiers
- Utiliser le champ group pour autoriser les accès à des membre d'un groupe

UPMC PARIS UNIVERSITAS

Sommaire

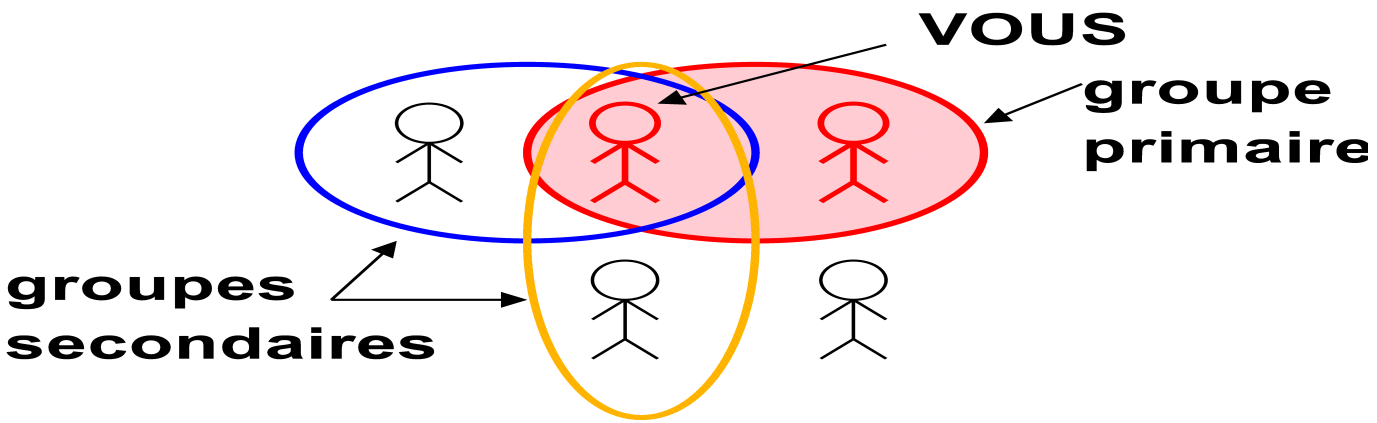
- Principe des droits d'accès
- `chmod`
- `umask`
- Droits spéciaux
- chown
- chgrp, newgrp
- Introduction aux ACL (Access Control List)

Droits d'accès

- Principe
- Sous Unix, l'accès aux objets est géré par un système de permissions
- Après authentification sur le système, l'utilisateur est identifié par un couple
  - « User ID » ou UID : identifiant utilisateur numérique unique
  - « Group ID » ou GID : identifiant du groupe primaire de l'utilisateur
- Les permissions permettent de définir des droits d'accès spécifiques à chaque utilisateur

Droits d'accès

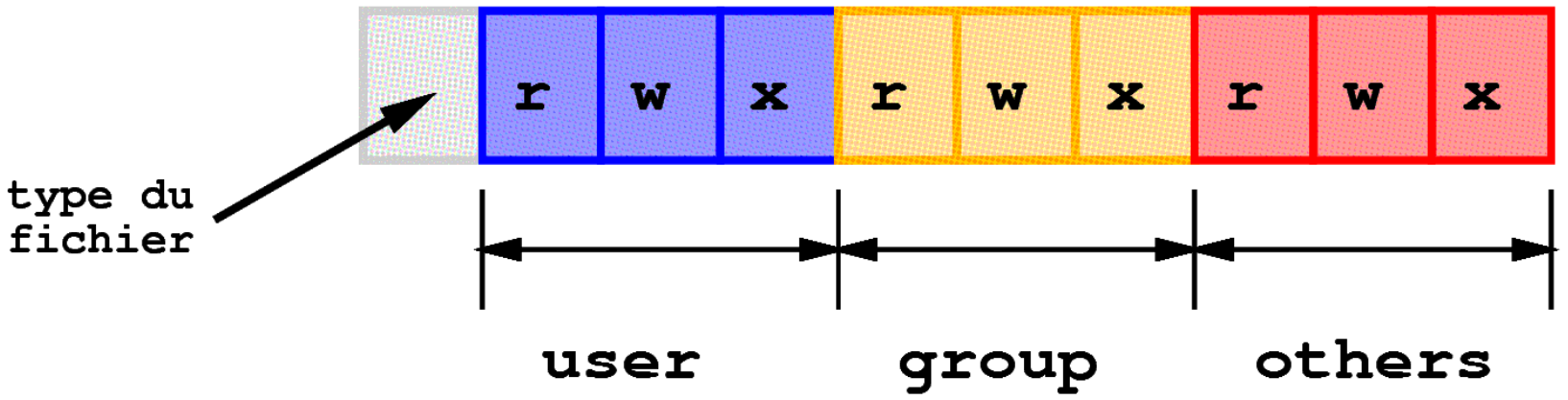
- A chaque objets, sont associés des droits d'accès à 3 catégories d'utilisateurs
    - l'utilisateur propriétaire (user)
    - le groupe propriétaire (group)
    - tous les autres (others)
- Un utilisateur appartient à un groupe primaire
- Un utilisateur peut appartenir à des groupes secondaires



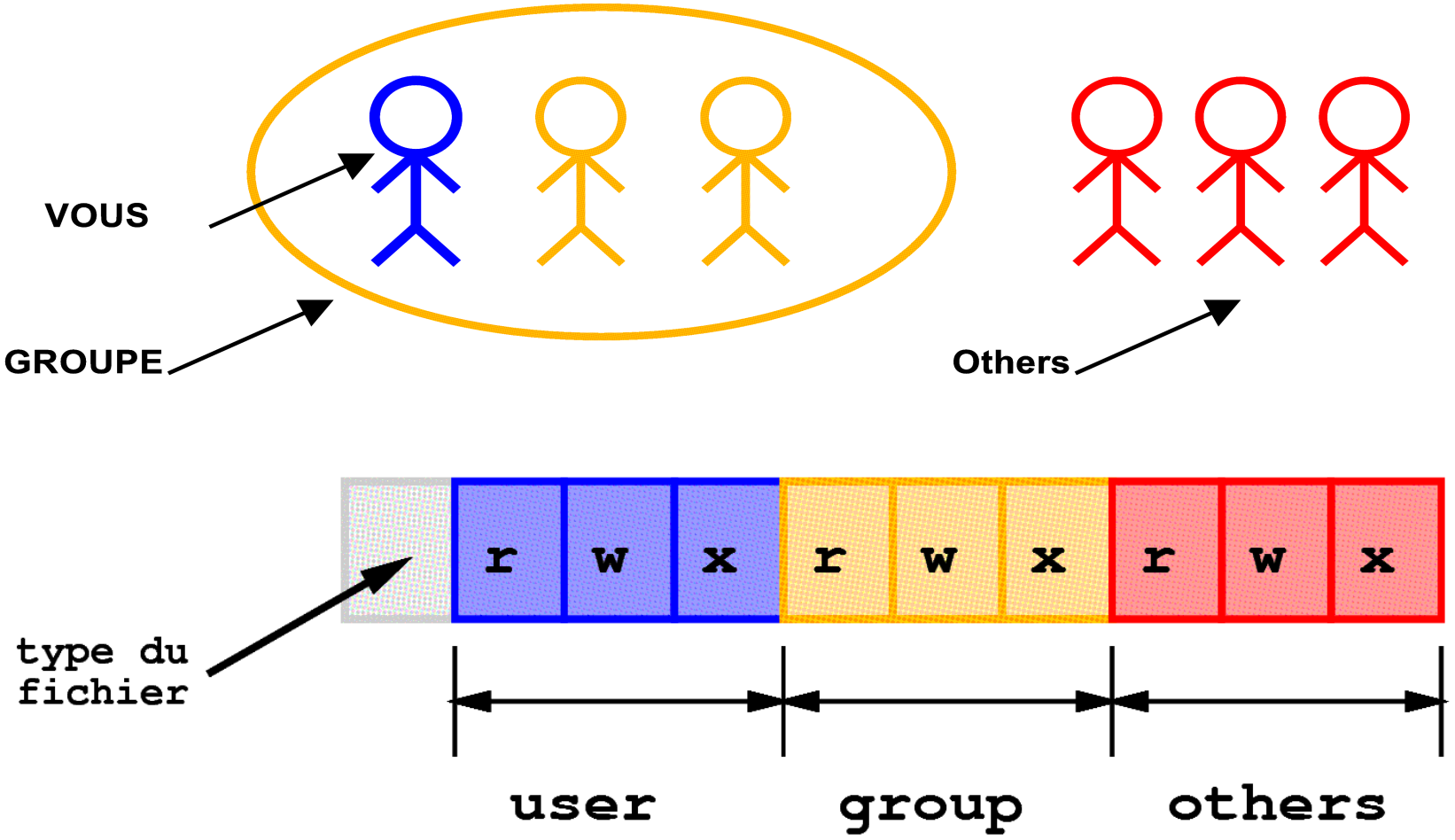
Droits d'accès

- Les droits d'accès sont stockés dans la structure inode du fichier
  - les droits sont visualisables par la commande « `ls -l` »
  - ils sont représentés par les 10 premiers caractères de la commande « `ls -l` »

```
[aoi@test ]$ ls -l c
-rw-r--r-- 3 franck franck 111 mar 13 19:34 c
```

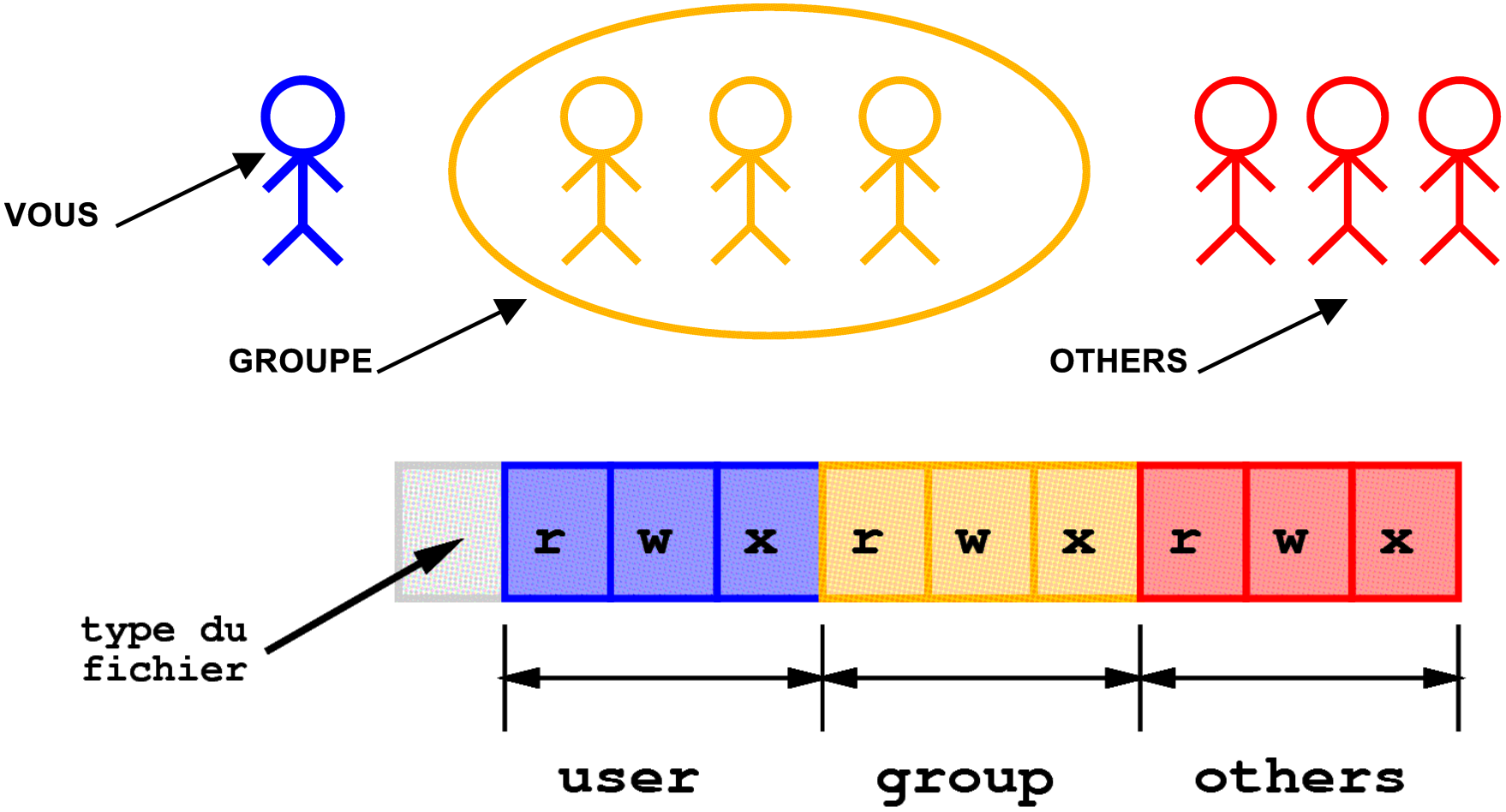


- Cas le plus courant
  - l'utilisateur appartient au groupe propriétaire



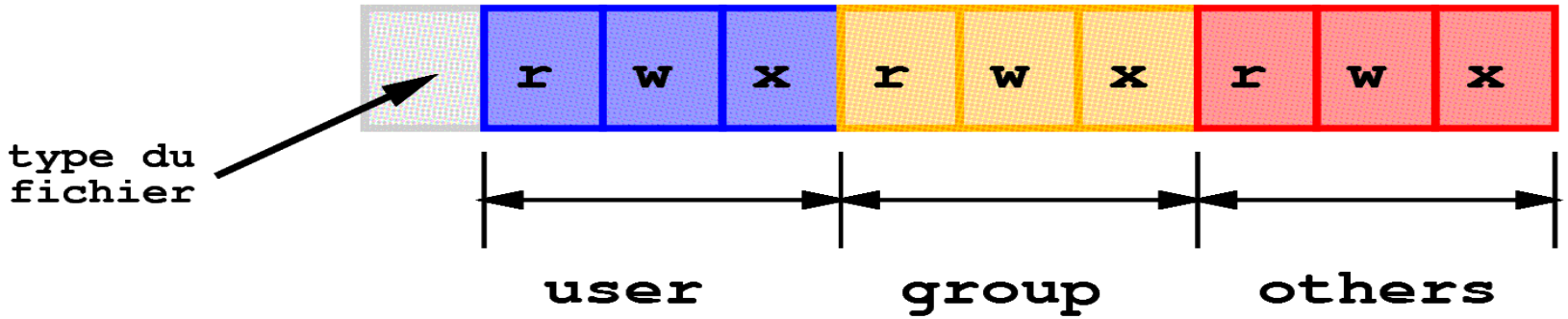
Droits d'accès

- Plus rarement mais possible
  - l'utilisateur n'appartient pas au groupe propriétaire



Droits d'accès

- Il existe 3 droits d'accès associés à chaque objet
  - droits du propriétaire (u – user)
  - droits des membres du groupe (g – group)
  - droits des autres utilisateurs (o- others)
- Il existe 3 types de permissions
  - droits en lecture (r - read)
  - droits en écriture (w – write)
  - droits en exécution (x – execute access)



Droits d'accès

- Dans le cas de fichiers
  - « r » : droit de lecture -> permet de lire ou copier le fichier
  - « w » : droit d'écriture -> permet d'ajouter, modifier ou supprimer des données du fichier
  - « x » : droit d'exécution -> permet d'exécuter le fichier (binaire ou script)

Droits d'accès

- Dans le cas de répertoires
  - « r » : permet de lister le contenu du répertoire
  - « w » : permet d'ajouter ou supprimer des données dans le répertoire.
    Ou plutôt permet de créer de modifier et supprimer des liens vers les inodes.
    Conséquence : possibilité de supprimer un fichier sur lequel l'utilisateur n'a aucun droit s'il a les droits d'écriture sur le répertoire
  - « x » : permet d'accéder aux fichiers du répertoire (droit de traversée). En gros toujours le bit x de positionné lors de la création de répertoire

Quelques règles

- Pour accéder à un fichier, il faut avoir le droit de franchissement de chacun des répertoires qui constituent le chemin
  - chaque répertoire du chemin doit posséder le droit x
- Pour écrire dans un fichier il faut avoir les droits d'écriture (w) sur le fichier (sauf root qui peut écrire et lire partout)
- Pour modifier les droits d'un fichier, il faut en être le propriétaire (ou être root)
- Pour créer, modifier, renommer, détruire un fichier, il faut avoir les droits d'écriture dans le répertoire contenant le fichier (toutes ces actions correspondent en définitive à des créations ou altérations de liens)
- root peut modifier les permissions de n'importe quel fichier

Quelques règles

- Détruire un fichier pour lequel on ne possède aucun droit !

```
# ll -d /reptest
drwxrwxrwx 2 root root 4096 2010-03-23 17:40 /reptest

 # touch /reptest/ficroot

 # chmod 700 /reptest/ficroot

# ll /reptest/ficroot
-rwx------ 1 root root 0 2010-03-23 17:41 /reptest/ficroot

$ id
uid=1000(franck) gid=1000(franck)
....

$ rm /reptest/ficroot
rm: détruire un fichier protégé en écriture fichier vide standard `/reptest/ficroot'? y

# ll /reptest/ficroot
ls: ne peut accéder /reptest/ficroot: Aucun fichier ou dossier de ce type
```

Quelques règles

- root peut écrire partout !

```
$ touch monfichier

$ ll monfichier
-rwx------ 1 franck franck 0 2010-03-23 17:47 monfichier

# id
uid=0(root) gid=0(root) groupes=0(root)

# echo "J'écris partout" > ~franck/test/monfichier

  $ cat monfichier
  J'écris partout
```

Types de fichiers

- Types de fichiers
  - « `-` » : fichier régulier (texte, exécutable, image,...)
  - « `d` » : répertoire (directory)
  - « `c` » : fichier associé à des périphériques en mode caractère (ex: liaison série)
  - « `b` » : fichier associé à des périphériques en mode bloc (disques, DVD,...)
  - « `l` » : liens symboliques
  - « `s` » : fichiers socket (named pipes adaptés au réseau)
  - « `p` » : named pipe; permet la communication entre deux processus (|)

UPMC PARIS UNIVERSITAS

Sommaire

- Principe des droits d'accès
- `chmod`
- `umask`
- Droits spéciaux
- chown
- chgrp, newgrp
- Introduction aux ACL (Access Control List)

Droits d'accès : chmod

- Commande « `chmod` » (change modes)
  - syntaxe : `chmod [options] modes objets`
  - spécification des modes sous 2 formes
    - forme symbolique
      rôle -> « u » : user; « g » (group); « o » (others); ou « a » (all)
      opérateur -> « + » (ajout); « - » (retrait); « = » (remplacement)
      permission -> « r » (read) ; « w » (write) ; « x » (execute)
      associations possibles en utilisant la virgule comme séparateur: u=rw,g=r
    - forme numérique
      les permissions sont exprimées en octal
      ex : `rwxr-xr-x` <=> 755

Droits d'accès : chmod

- Permissions supplémentaires
  - X : execution seulement si le fichier est un répertoire ou qu'il possède déjà au moins une permission d'exécution
  - s : SUID ou SGID
  - t : sticky bit
  - u : toutes les permissions existantes du propriétaire
  - g : toutes les permissions existantes du groupe
  - o : toutes les permissions existantes pour tout le monde

Droits d'accès : chmod

- Options :
  - -R : modification récursive
  - -v : mode verbeux
  - -c : idem -v mais n'affiche que les modifications
  - -f : mode silencieux

Droits d'accès : chmod

- Qui a le droit de changer des permissions ?
  - root
  - Le propriétaire du fichier
- Attention : un droit d'écriture sur un répertoire ne donne pas forcément le droit de modifier les permissions des fichiers qu'il contient
  - Les permissions sont définies au niveau de la structure inode du fichier
  - Le droit de modification d'un répertoire n'agit que sur les liens qui pointent sur l'inode

Droits d'accès : chmod

- 9ème bit pour définir les permissions spéciales
  - 3 valeurs
  - 4 : SUID
  - 2 : GUID
  - 1 : sticky bit
- Pour définir 2 permissions spéciale, calcul identique aux permissions standard
  - s : SUID ou SGID
  - t : sticky bit
  - u : toutes les permissions existantes du propriétaire
  - g : toutes les permissions existantes du groupe
  - o : toutes les permissions existantes pour tout le monde

Droits d'accès : chmod

Droits	Valeur en octal
`---`	0
`--x`	1
`-w-`	2
`-wx`	3
`r--`	4
`r-x`	5
`rw-`	6
`rwx`	7

- Exemples
  - 755 = `rwx r-x r-x`
  - 644 = `rw- r-- r--`
  - 600 = `rw- --- ---`

Droits d'accès : chmod

```
[aoi@test]$ ls -l fichier.txt
-rw-r--r-- 1 franck franck 259 mar 21 21:57 fichier.txt

[aoi@test]$ chmod g+w fichier.txt
[aoi@test]$ ls -l fichier.txt
-rw-rw-r-- 1 franck franck 259 mar 21 21:57 fichier.txt

[aoi@test]$ chmod o=rw fichier.txt
[aoi@test]$ ls -l fichier.txt
-rw-rw-rw- 1 franck franck 259 mar 21 21:57 fichier.txt

[aoi@test]$ chmod 640 fichier.txt
[aoi@test]$ ls -l fichier.txt
-rw-r----- 1 franck franck 259 mar 21 21:57 fichier.txt

[aoi@test]$ chmod 777 fichier.txt
[aoi@test]$ ls -l fichier.txt
-rwxrwxrwx 1 franck franck 259 mar 21 21:57 fichier.txt*
```

UPMC PARIS UNIVERSITAS

Sommaire

- Principe des droits d'accès
- `chmod`
- `umask`
- Droits spéciaux
- chown
- chgrp, newgrp
- Introduction aux ACL (Access Control List)

Droits d'accès : umask

- Commande « `umask` » (user mask)
  - droits par défaut à la création des objets
  - syntaxe : `umask [modes]`
  - `-S` : affiche le masque sous forme symbolique
  - modes : indique en octal les bits qui ne seront pas positionnés
  - Le calcul se base sur 777 pour les répertoires et exécutables et 666 pour tous les autres fichiers

Droits	Valeur en octal
`---`	7
`--x`	6
`-w-`	5
`-wx`	4
`r--`	3
`r-x`	2
`rw-`	1
`rwx`	0

Droits d'accès : umask

- Dans certaine distribution umask est définit dans /etc/profile (Debian/Ubuntu)
- `umask` est définit pour les distributions RH/Centos/Fedora dans le fichier `/etc/bashrc`

```
# By default, we want this to get set.
# Even for non-interactive, non-login shells.
if [ $UID -gt 99 ] && [ "`id -gn`" = "`id -un`" ]; then
        umask 002
else
        umask 022
fi
```

```
[franck@localhost ~]$ su -
Mot de passe :
[root@localhost ~]# id
uid=0(root) gid=0(root) groupes=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
[root@localhost ~]# umask
0022
```

Droits d'accès : umask

0002 correspond à :
rwx rwx r-x pour un binaire ou un répertoire
rw- rw- r-- pour un fichier texte

```
[aoi@test]$ umask
0002

[aoi@test]$ touch fichier2.txt

[aoi@test]$ ls -l fichier2.txt
-rw-rw-r-- 1 franck franck 0 mar 21 22:17 fichier2.txt

[aoi@test]$ umask 0000

[aoi@test]$ touch fichier3.txt

[aoi@test]$ ls -l fichier3.txt
-rw-rw-rw- 1 franck franck 0 mar 21 22:18 fichier3.txt
```

UPMC PARIS UNIVERSITAS

Sommaire

- Principe des droits d'accès
- `chmod`
- `umask`
- Droits spéciaux
- chown
- chgrp, newgrp
- Introduction aux ACL (Access Control List)

Droits spéciaux

- bit setuid (4000 en octal)
  - programme exécuté avec les droits de l'utilisateur propriétaire

```
[aoi@test]$ chmod u+sx prog
[aoi@test]$ ls -l prog
-rwsrw-rw- 1 franck franck 0 mar 21 22:30 prog*
```

```
[aoi@test]$ ls -l /usr/bin/passwd
-r-s--x--x 1 root root 15552 mai  8  2006
/usr/bin/passwd*
```

- bit setgid (2000 en octal)
  - programme exécuté avec les droits du groupe propriétaire

```
[aoi@test]$ chmod g+sx prog2
[aoi@test]$ ls -l prog2
-rw-rwsrw- 1 franck franck 0 mar 21 23:01 prog2*
```

Droits spéciaux

- sticky bit setuid (1000 en octal)
  - Lorsque ce bit est positionné, on ne peut effacer que les fichiers dont on est propriétaire
  - exemple : `/tmp` (répertoire de stockage temporaire)

```
[aoi@test]$ ls -ld /tmp
drwxrwxrwt 15 root root 7168 mar 21 22:58 /tmp/
[aoi@test]$ touch /tmp/monfic
[aoi@test]$ ls -l /tmp/monfic
-rw-rw-rw- 1 franck franck 0 mar 21 23:15
/tmp/monfic
```

```
[mc2@localhost ~]$ id
uid=503(mc2) gid=503(mc2) groupes=503(mc2)
[mc2@localhost ~]$ ls -l /tmp/monfic
-rw-rw-rw- 1 franck franck 0 mar 21 23:15 /tmp/monfic
[mc2@localhost ~]$ rm /tmp/monfic

rm: ne peut enlever `/tmp/monfic': Opération non permise
```

UPMC PARIS UNIVERSITAS

Sommaire

- Principe des droits d'accès
- `chmod`
- `umask`
- Droits spéciaux
- chown
- chgrp, newgrp
- Introduction aux ACL (Access Control List)

- `chown [-Rh] user objet`
  - change le propriétaire des objets
  - option « `-R` » : changement résursif dans une arborescence
  - option « `-h` » : en cas de lien symbolique, change le propriétaire du lien et non les objets pointés par le lien
  - commande réservée à root (sinon problème de sécurité)

```
[franck@localhost UNIX]$ ls -l result.txt
-rw-r--r-- 1 franck franck 259 mar 28 23:27 result.txt
[franck@localhost UNIX]$ su
Mot de passe :
[root@localhost UNIX]# chown mc2test result.txt
[root@localhost UNIX]# ls -l result.txt
-rw-r--r-- 1 mc2test franck 259 mar 28 23:27 result.txt
```

- `chgrp [-Rh] groupe objet`
  - change le groupe primaire des objets
  - fonctionnement identique à `chown`

```
[franck@localhost UNIX]$ ls -l result.txt
-rw-r--r-- 1 mc2test franck 259 mar 28 23:27 result.txt

[franck@localhost UNIX]$ su
Mot de passe :

[root@localhost UNIX]# chgrp mc2test result.txt
[root@localhost UNIX]# ls -l result.txt
-rw-r--r-- 1 mc2test mc2test 259 mar 28 23:27 result.txt
```

Gestion des utilisateurs et groupes

- Faire appartenir un fichier à un groupe
  - `chrgrp <groupe> <fichiers>`
  - `chown user:groupe <fichiers>`
  - Pour que cela fonctionne : il faut être root ou appartenir au groupe

[franck@localhost ~]$ ls -l fictest
-rw-rw-r-- 1 franck **franck** 0 mar 14 11:45 fictest

[franck@localhost ~]$ id
uid=500(franck) gid=500(franck) groupes=500(franck),**503(usertest)**

[franck@localhost ~]$ **chgrp usertest fictest**

[franck@localhost ~]$ ls -l fictest
-rw-rw-r-- 1 franck **usertest** 0 mar 14 11:45 fictest

- Changer les permissions de groupe d'un fichier (cf. plus haut)
  - `chmod`

- Permissions spéciales pour un groupe
  - *setgid bit* s'il s'agit d'un fichier
    S'il est positionné pour un fichier exécutable, le processus lancé aura l'identifiant de groupe du fichier à la place de celui de l'utilisateur
  - *setgid bit* s'il s'agit d'un répertoire
    les fichiers crées dans le répertoire auront pour groupe celui du répertoire et non celui de l'utilisateur qui les crée

[franck@localhost ~]$ ls -l fichier
-rw-rw-r-- 1 franck franck 0 mar 14 12:03 fichier

[franck@localhost ~]$ **chmod g+xs fichier**

[franck@localhost ~]$ ls -l fichier
-rw-rw**s**r-- 1 franck franck 0 mar 14 12:03 fichier

[franck@localhost ~]$ ls -ld repertoire
drwxrwxr-x 2 franck franck 4096 mar 14 12:09 repertoire

[franck@localhost ~]$ **chmod g+s repertoire**/

[franck@localhost ~]$ ls -ld repertoire
drwxrw**s**r-x 2 franck franck 4096 mar 14 12:09 repertoire

[franck@localhost ~]$ ls -ld repertoire
drwxrw**s**r-x 2 franck franck 4096 mar 14 12:09 repertoire

[franck@localhost ~]$ touch repertoire/fic1

[franck@localhost ~]$ ls -l repertoire/fic1
-rw-rw-r-- 1 franck **franck** 0 mar 14 12:12 repertoire/fic1

[franck@localhost ~]$ **chgrp usertest repertoire**/

[franck@localhost ~]$ ls -ld repertoire
drwxrwsr-x 2 franck **usertest** 4096 mar 14 12:12 repertoire

[franck@localhost ~]$ touch repertoire/fic2

[franck@localhost ~]$ ls -l repertoire/fic2
-rw-rw-r-- 1 franck **usertest** 0 mar 14 12:14 repertoire/fic2

- Modifier temporairement le groupe primaire
  - `newgrp <groupe>`
  - Equivalent à la commande su pour les groupes
  - `newgrp - <groupe>` : l'option « - » réinitialise l'environnement utilisateur
  - Permet de changer de groupe primaire en lançant un nouveau shell

```
[franck@localhost ~]$ id
uid=500(franck) gid=500(franck) groupes=500(franck),503(usertest)
[franck@localhost ~]$ newgrp usertest
[franck@localhost ~]$ id
uid=500(franck) gid=503(usertest) groupes=500(franck),503(usertest)
[franck@localhost ~]$ exit
exit
[franck@localhost ~]$ id
uid=500(franck) gid=500(franck) groupes=500(franck),503(usertest)
```

UPMC PARIS UNIVERSITAS

Sommaire

- Principe des droits d'accès
- `chmod`
- `umask`
- Droits spéciaux
- chown
- chgrp, newgrp
- Introduction aux ACL (Access Control List)

ACL

- Limitation des droits classiques
  - on ne peut indiquer des droits que sur l'utilisateur propriétaire, un seul groupe propriétaire, le « reste du monde »
  - dans le cas de données accédées par un nombre important d'utilisateurs avec des types d'accès différents (travail collaboratif) : ce n'est pas suffisant
- Solution : les ACL (Access Control List)
  - extension du nombre des utilisateurs et des groupes
  - les implémentations des ACL sur les différents Unix ne sont pas forcément compatibles entre elles
  - sous Linux, il faut :
    - un noyau 2.6.x ou 2.4.x (patché)
    - un filesystem compatible (ext2, ext3, jfs, xfs, resiserfs, nfs)

- Exemple
  - un répertoire compta; un groupe « compta1 » avec des droits de consultation uniquement; un groupe « compta2 » avec des droits de modification

```
[aoi@test]$ ll -d dsk/compta/
drwx------ 2 franck franck 1024 mar 24 09:25 dsk/compta//

[aoi@test]$ ll -d dsk/compta/
drwxrw----+ 2 franck franck 1024 mar 24 09:25 dsk/compta//
[aoi@test]$ getfacl dsk/compta//
# file: dsk/compta
# owner: franck
# group: franck
user::rwx
group::---
group:compta1:r--
group:compta2:rw-
mask::rw-
other::--
```

mise en place des ACL

- Vérification au niveau du noyau

```
[aoi@test]$ grep ACL /boot/config-2.6.17-6mdv
# CONFIG_RSBAC_ACL is not set
CONFIG_EXT2_FS_POSIX_ACL=y
CONFIG_EXT3_FS_POSIX_ACL=y
CONFIG_REISERFS_FS_POSIX_ACL=y
CONFIG_JFS_POSIX_ACL=y
CONFIG_FS_POSIX_ACL=y
CONFIG_XFS_POSIX_ACL=y
CONFIG_TMPFS_POSIX_ACL=y
CONFIG_NFS_V3_ACL=y
CONFIG_NFSD_V2_ACL=y
CONFIG_NFSD_V3_ACL=y
CONFIG_NFS_ACL_SUPPORT=m
CONFIG_GENERIC_ACL=y
```

- Si le noyau ne supporte pas les ACL : il faut recompiler le noyau

mise en place des ACL

- Au montage du système de fichier, il faut indiquer la prise en charge des ACL (les commandes suivantes seront détaillées plus loin dans le cours)

```
[aoi@test]$ dd if=/dev/zero of=/tmp/exemple.raw bs=1M count=128
....
134217728 octets (134 MB) copiés, 1,03855 seconde, 129 MB/s

[aoi@test]$ /sbin/mkfs.ext3 /tmp/exemple.raw
....

[root@localhost UNIX]# mount -o loop,acl -t ext3 /tmp/exemple.raw
~franck/UNIX/dsk/

 [root@localhost UNIX]# mount
 ...
 /tmp/exemple.raw on /home/franck/UNIX/dsk type ext3
 (rw,loop=/dev/loop0,acl)
```

- Exemple
  - un répertoire compta; un groupe « compta1 » avec des droits de consultation uniquement; un groupe « compta2 » avec des droits de modification

```
[aoi@test]$ ll -d dsk/compta/
drwx------ 2 franck franck 1024 mar 24 09:25 dsk/compta//

[aoi@test]$ ll -d dsk/compta/
drwxrw----+ 2 franck franck 1024 mar 24 09:25 dsk/compta//
[aoi@test]$ getfacl dsk/compta//
# file: dsk/compta
# owner: franck
# group: franck
user::rwx
group::---
group:compta1:r--
group:compta2:rw-
mask::rw-
other::--
```

gestion des ACL

- Il existe deux commandes spécifiques pour gérer les ACL
  - Commande : `setfacl options droits fichiers`
  - `setfacl` permet de modifier les ACL ainsi que les droits classiques
  - options
    - « -m » : pour modifier des permissions
    - « -x » : pour supprimer des permissions
  - droits
    - « u: », « g: », « o: » : définissent la portée des droits pour user, group et others
    - « r », « w », « x » : définissent les permissions classiques

gestion des ACL

- Il existe deux commandes spécifiques pour gérer les ACL
  - Commande : `getfacl fichiers`
  - `getfacl` affiche les ACL ainsi que les droits classiques

- Exemple

```
[aoi@test]$ touch datacpt
[aoi@test]$ ls -l datacpt
-rw-r--r-- 1 franck franck 0 mar 24 17:18 datacpt

[root@localhost UNIX]# setfacl -m group:compta1:r-- datacpt
[root@localhost UNIX]# setfacl -m group:compta2:rw- datacpt
[root@localhost UNIX]# setfacl -m user:mc2test:rw- datacpt

[aoi@test]$ getfacl datacpt
# file: datacpt
# owner: franck
# group: franck
user::rw-
user:mc2test:rw-
group::r--
group:compta1:r--
group:compta2:rw-
mask::rw-
other::r--
```

gestion des ACL

- Il n'y a pas d'héritage par défaut des ACL par les objets enfants (fichiers d'un répertoire)
- Si besoin, il faut le définir explicitement avec l'option « d: »

```
[root@localhost UNIX]# setfacl -m user:mc2test:rw- rep
[root@localhost UNIX]# setfacl -m group:compta2:rw- rep

[aoi@test]$ touch rep/fichier
[aoi@test]$ getfacl rep/fichier
# file: rep/fichier
# owner: franck
# group: franck
user::rw-
group::r--
other::r--
```

Pas d'héritage des ACL du répertoire rep

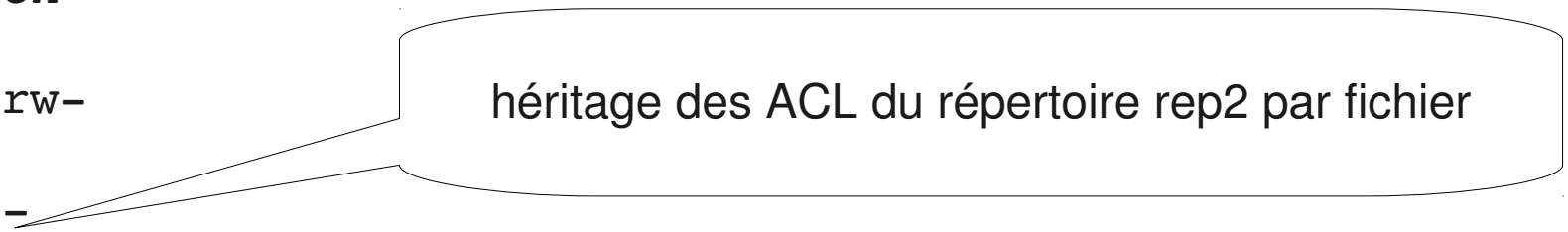
```
[aoi@test]$ ls -l rep/fichier
-rw-r--r-- 1 franck franck 0 mar 24 17:44 rep/fichier
```

- En utilisant l'option « d: »

```
[root@localhost UNIX]# setfacl -m d:group:compta2:rw- rep2
[root@localhost UNIX]# setfacl -m d:user:mc2test:rw- rep2

[aoi@test]$ touch rep2/fichier
[aoi@test]$ getfacl rep2/fichier
# file: rep2/fichier
# owner: franck
# group: franck
user::rw-
user:mc2test:rw-
group::r-x
#effective:r--
group:compta2:rw-
mask::rw-
other::r--

[aoi@test]$ ls -l rep2/fichier
-rw-rw-r--+ 1 franck franck 0 mar 24 17:48 rep2/fichier
```



- Compatibilité des commandes
  - Sous Linux, il faut vérifier si les commandes de manipulation de fichier supportent les ACL ou utiliser l'option adéquate
    - `cp -a` : pour conserver les attributs
    - `mv` : conserve les attributs par défauts
    - `tar` : ne conserve pas les attributs. Utiliser `star` à la place
  - Sous Solaris, les commandes de manipulation de fichier supportent les ACL

Quelques conseils LPI...

- Connaître l'utilisation des commandes décrites
- Connaître l'utilisation de l'option -g pour que ces commandes agissent sur les quotas de groupe