ACLs

Pé-requis :

a. toujours dans le dossier "~/test" utilisé dans les chapitres précédents. Cette fois, nous mettrons tout à 770 root root :

```
# chmod -R 0770 test
# chmod -R u-s,g-s test
# chown -R root:root test
# setfacl -R -b test
# chmod 777 /home/<votre utilisateur>
```

b. Puis nous ajoutons le droit d'execution à tout le monde sur le fichier "test/monscript.sh"

```
# chmod +x test/monscript.sh
```

c. Créer l'utilisateur "toto", ayant "users" pour groupe principal et "root" pour groupe secondaire.

```
# useradd -g users -G root toto
```

d. Bien entendu, vérifiez également tout les pré-requis pour les acls (kernel, paquet et option de montage).

Excercices :

1) afficher les ACLs de "test" puis de "test/fichier1"

```
$ getfacl test/fichier1
# file: fichier1
# owner: root
# group: root
user::rwx
group::rwx
other::rwx
```

2) appliquer les ACLs suivantes et, à chaque étape, lancer les tests présentés ci-dessous :

- sur le dossier test et sur le fichier "test/fichier1" : (bien entendu, vous devez réeffectuer l'opération a des pré-requis entre chaque opération, hormis entre l'étape d et e)
- a) autoriser votre utilisateur (nmace) à lire

```
# setfacl -m u:nmace:r test
```

- b) ajouter au groupe users tous les droits

```
# setfacl -m g:users:rwx test
```

Tout nos utilisateurs ont tout les droits, étant donné qu'ils appartiennent à "users" et que l'acl l'emporte sur le reste. Bien entendu, si vous aviez laissé l'acl créée en a., l'utilisateur nmace n'aurait eu que le droit de lire.

- c) faites en sorte que le groupe propriétaire n'ai plus aucun droit

```
# setfacl -m g::--- test
```

Ainsi, toto membre du groupe root perd ses privilèges. Cela ne serait pas vrais si une règle spécifique à toto existait.

- d) donnez tout les droits au groupe "users" et à l'utilisateur "nmace"

```
# setfacl -m u:nmace:rwx,g:users:rwx test
```

dans les 3 cas, ces utilisateurs ont donc tout les droits (une acl spécifique l'emporte sur les acl généralistes / droits posix)

- e) établissez un masque à "r--"

```
# setfacl -m m::r test
```

tout le monde (hormis l'utilisateur propriétaire pour qui le masque ne s'applique pas) n'a que le droit de lecture sur les dossiers et fichiers

Tests :

1. en tant que votre utilisateur (nmace) :
   - afficher le contenu du dossier "test"
   - afficher celui du fichier "test/fichier1
   - lancer la commande suivante : $ echo "test X" >> test/fichier1
   - créer un fichier "toto" dans "test", puis le supprimer si cela a fonctionné
   - déplacez vous dans "test"
   - executez le script monscript.sh (voir les TPs précédents)
   - 2. effectuer les mêmes opérations en tant que toto
   - 3. effectuer les mêmes opérations en tant que toto après avoir lancé la commande suivante : # usermod -G "" toto (pensez à effectuer la commande inverse avant de passer à l'étape suivante # usermod -G "root" toto)

Pour la correction des tests, voici un petit script :

========== ex55-test.sh ==========

```
#!/bin/bash

## TODO : suppression du fichier de logs

### importations

cd `dirname $0`

# libraire de présentation (couleurs, cecho, indentation ...)
. presentation.lib.sh
. entrees.lib.sh

### déclariation des variables

testdirname="test"
rep="/home/drno/"
testdir="${rep}${testdirname}"
script="${testdir}/monscript.sh"
user1="nmace"
user2="toto"
logs="/tmp/$0.logs"
fichier1=$testdir"/fichier1"

# début et fin de la commande echo en rouge, utilisée pour la variable mon test
```

```
#+ permettra d'afficher les messages informant des différentes étapes dans une couleur distinctive,
#+ afin de les différencier des messages d'erreur ou de réussite de chaque commande
echo_debut='echo -e "'$rougefonce
echo_fin=$neutre""'

awk_cmd_debut='/'
awk_cmd_fin='/  { print $1 " " $3 " " $4 " " $9 }'
# awk '/test/{print $1 " " $3 " " $4 " " $9}
awk_cmd_1=$awk_cmd_debut"$testdirname\$"$awk_cmd_fin
awk_cmd_2=$awk_cmd_debut'$fichier1'$awk_cmd_fin



# variable contenant les différentes commandes devant être executées par su monuser -c
#+ (donc en tant que l'utilisateur monuser)
#+ il s'agit simplement de la restrancription des tests
montest="$echo_debut ls $testdir $echo_fin ;
    ls $testdir > /dev/null && echo ok || echo -n 'erreur';
    $echo_debut cat '$testdir/fichier1' $echo_fin ;
    cat $testdir/fichier1 > /dev/null && echo ok || echo -n 'erreur' ;
    $echo_debut écriture dans fichier1 $echo_fin ;
    echo test $TPCMPT >> $testdir/fichier1 && echo ok || echo -n 'erreur';
    $echo_debut création et suppression du fichier $testdir/toto $echo_fin ;
    if [ -e $testdir/toto ]; then
    echo le fichier $testdir/toto existe déjà ;
            rm $test/toto && echo et a pu être supprimé ;
    else
    ( touch $testdir/toto && rm $testdir/toto ) 2>&1 && echo ok || echo -n 'erreur' ;
    fi ;
    $echo_debut déplacement dans $testdir $echo_fin ;
    cd $testdir && echo ok || echo 'erreur';
    $echo_debut lancement du script $script $echo_fin ;
    $script > /dev/null && echo ok || echo 'erreur';

    exit 0 ;"

## définition des fonctions

# fonction d'initalisation de notre script
debut()
{
    # si le fichier de logs n'existe pas, alors écrire une première phrase dedans
    if [ ! -e $logs ]; then
    echo "LPIC1 - TPs - 5.5 - ACLs : résultat du script" > $logs
    fi

    # si la variable d'environnement TPCCMPT n'a pas de valeur (a une longueur nulle) alors
    #+ l'initialiser à 1 et l'exporter
    if [ -z $TPCMPT ]; then
```

```
    TPCMPT=1
    export TPCMPT
    fi

    # supprimer le fichier toto si il existe
    if [ -e $testdir/toto ]; then
    rm toto
    fi

    # créer le fichier de script si il n'existe pas
    if [ ! -e $script ]; then
    echo -e '#!/bin/bash\necho youpi' > $script
    chmod +x $script
    fi;

    # créer l'utilisateur $user2 si il n'existe pas, sinon, le modifier
    # TODO : tester uniquement pour la colonne nom
    if [ `grep -c $user2 /etc/passwd` == 0 ]; then
    useradd -m -g "users" -G "root" -s /bin/bash $user2
    else
    usermod -g "users" -G "root" -s /bin/bash $user2
    fi;
}



# première étape de notre script, permettant d'afficher les droits sur les fichiers
afficher_droits()
{
    cecho $vertfonce "Droits établis :"
    cecho $rougefonce "répertoire de travail :" | indenter
    getfacl -pcE $rep | indenter
    ls -l $rep | awk "$awk_cmd_1"
    getfacl -pcE $testdir | indenter
    ls -l $testdir | awk "$awk_cmd_2"
    getfacl -pcE $fichier1 | indenter
}

# méthode principale de notre script, effectant le test pour chaque cas de figure
main()
{
    echo
    cecho $vertfonce "1) en tant que ${user1}"
    su $user1 -c "$montest" | indenter
    echo
    cecho $vertfonce "2) en tant que $user2 avec root comme groupe secondaire"
    su $user2 -c "$montest" | indenter
    usermod -G "" $user2
```

```
	echo
	cecho $vertfonce "3) en tant que $user2 n'appartenant pas au groupe root"
	su $user2 -c "$montest" | indenter
	usermod -G "root" $user2
}


## corps du script

# appel des différentes fonctions, et enregistrement de leurs sorties dans le fichier de log
{
afficher_droits
debut
main
} &> $logs

# enfin, nous affichons ce fichier de logs
less -R $logs

if read_boolean "supprimer le fichier de log ? (y/N)"; then
	rm $logs && echo -e "fichier supprimé"
fi;

exit 0

================ presentation.lib.sh ========================

## variables

# couleurs pour echo -e et cecho()
noir='\e[0;30m'
gris='\e[1;30m'
rougefonce='\e[0;31m'
rose='\e[1;31m'
vertfonce='\e[0;32m'
vertclair='\e[1;32m'
orange='\e[0;33m'
jaune='\e[1;33m'
bleufonce='\e[0;34m'
bleuclair='\e[1;34m'
violetfonce='\e[0;35m'
violetclair='\e[1;35m'
cyanfonce='\e[0;36m'
cyanclair='\e[1;36m'
grisclair='\e[0;37m'
blanc='\e[1;37m'
neutre='\e[0;m'
```

```
## fonctions

# indenter chaque ligne du fichier passé en argument
# usage : indenter ls -l
indenter()
{
#    for ligne in $@ ;
     while read ligne;
     do
     echo -e "\t$ligne"
     done
}

# afficher un texte dans la couleur donnée
# usage : cecho $couleur "texte à afficher"
# où $couleur est le valeur d'une des variables de couleur définie dans cette librairie
# attention, le texte à afficher ne doit pas contenir de retour à la ligne
cecho()
{
     echo -e "$1$2$neutre"
}

====================== entrees.lib.sh ==================

#!/bin/bash

read_boolean()
{
     caseglob1="[oO]?([uU])?([iI])"
     caseglob2="[yY]?([eE])?([sS])"
     shopt -s extglob
     read -p "$1" stringin
#    first_char=${$stringin:0:1}
     echo -n "$stringin" | wc -c
     case $stringin in
     ( $caseglob1|$caseglob2 )
               return 0 ;
               ;;
     (*)
               return 1 ;
               ;;
     esac
}

#: ${1?"Usage: $0 ARGUMENT"}
```

UMASK : droits par defaut

Cette valeur par défaut signifie que les droits de tout nouveau fichier (créé avec la commande touch, disons) seront masqués par ce nombre.
Dans le cas d'un fichier exécutable ou dossier, les droits seront fixés comme 777-022=755, c'est-à-dire -rwxr-xr-x.

Dans le cas d'un fichier non-exécutable (texte, image, etc.), les droits seront 666-022=644 soit -rw-r-.-r-.-

Le masque 022 exlut les permissions d'écriture pour le groupe (g) et les autres (o).
Un umask de 026 exclurait les droits de lecture et l'écriture pour les autres (o) mais permettrait au groupe (g) la lecture et l'écriture.]

Lorsque vous créez un fichier ou un répertoire, les autorisations assignées par défaut au fichier ou répertoire sont contrôlées par le **masque utilisateur**. Le masque utilisateur est défini par la commande umask dans un fichier d'initialisation utilisateur. Vous pouvez afficher la valeur courante du masque utilisateur en tapant umask et en appuyant sur Entrée.
Le masque utilisateur contient les valeurs octales suivantes :

- Le premier chiffre définit les autorisations pour l'utilisateur.
- Le deuxième chiffre définit les autorisations pour le groupe.
- Le troisième chiffre définit les autorisations pour les autres, aussi appelé world.

Notez que si le premier chiffre est zéro, il n'est pas affiché. Par exemple, si le masque utilisateur est défini sur 022, le nombre 22 s'affiche.

Pour déterminer la valeur umask à définir, soustrayez la valeur des autorisations souhaitées de 666 (pour un fichier) ou 777 (pour un répertoire). Le résultat de cette soustraction correspond à la valeur à utiliser avec la commande umask. Par exemple, supposons que vous souhaitez modifier le mode par défaut pour les fichiers et le passer à 644 (rw-r--r--). La différence entre 666 et 644 est 022, ce qui correspond à la valeur à utiliser en tant qu'argument de la commande umask.

Vous pouvez également déterminer la valeur umask que vous voulez définir à l'aide du tableau suivant. Ce tableau indique les autorisations de fichiers et de répertoires qui sont créées pour chacune des valeurs octales de umask

Valeur octale `umask`	Autorisations de fichier	Autorisations de répertoire
0	`rw-`	`rwx`
1	`rw-`	`rw-`
2	`r--`	`r-x`
3	`r--`	`r--`
4	`-w-`	`-wx`
5	`-w-`	`-w-`
6	`--x`	`--x`
7	`---` (aucune)	`---` (aucune)

Exercice umask

1) afficher votre umask

a. en octal `umask -p`
b. de manière "classique" `umask -S`

2) modifiez votre umask pour obtenir les résultats suivants, en octal puis avec la syntaxe "classique". Vous pouvez créer des fichiers et dossiers à chaque étape dans votre dossier "test" afin de vérifier le résultat

a. tous les droits à la création (aussi bien pour les fichiers que les dossiers)
`#umask 0000`
b. droit de lecture pour tout le monde, droit d'execution pour tout le monde sur les dossiers, et seul le propriétaire aura le droit d'écriture `#umask 0022`
c. droit de lecture et d'écriture pour l'utilisateur et le groupe propriétaire `#umask 0067`
d. aucun droit (execution sur les dossiers comprise) pour tout autre utilisateur que le propriétaire
`#umask 0077`

3) changez d'utilisateur grâce à la commande su, et afficher son masque
4) faire de même mais avec la commande su -

srazat@debian-stef:/home/stephane$ umask -S
u=rwx,g=rwx,o=rwx
srazat@debian-stef:/home/stephane$ su -
Mot de passe :
root@debian-stef:~# umask -S
u=rwx,g=rwx,o=rwx
root@debian-stef:~#

5) ouvrir un autre terminal, avec le même utilisateur, et afficher le umask 0777
a. à partir de la ligne de commande (donc du terminal déjà lancé) 0777
b. à partir de l'environnement de bureau (donc totalement indépendamment du terminal précédent) 0222

6) enfin, changer graphiquement d'utilisateur 0222

7) redémarrez votre machine et afficher le umask

Bonus 1) trouver une solution pour faire en sorte que le umask soit définitivement modifié pour tous les utilisateurs > **`modifier le fichier #vim /etc/login.defs`**

Bonus 2) de même mais pour un utilisateur uniquement
Edition du fichier de conf utilisateur ~/ **.profile** enter 000 par exemple, fermer la session, réouvrir la session. Pour vérifier umask