3R-IN3 – Systèmes d'exploitation
TD1

Xavier Hilaire
x.hilaire@esiee.fr

14 mars 2025

1 Droits d'accès

L'utilisateur `fred`, de groupe `fred`, lance une commande `ls -l` dans un terminal et obtient la sortie suivante :

```
$ ls -l
total 316
drwxr-xr-x 4 fred fred   4096 29 sept.  2020  Calltech101
-rwxrwxr-x 1 fred fred    531 28 nov.   2011  cent_moment.m
-rwxr--r-x 1 eric fred   1463 17 sept.  2018  evalBOF
-rw-r--r-- 1 fred fred   1713 26 sept.  2018  #evalBVF.m#
-rw-r-xr-- 1 fred info   1749 27 sept.  2018  evalBVF
-rw-rwxr-x 1 fred fred   1586 28 nov.   2011  feature_vec.m
-rwsrws--x 1 eric info  27331 28 nov.   2011  feature_vec
-rw-rw---- 1 fred fred   4660 28 nov.   2011  feature_res
-rw-r--r-- 1 fred fred   8358 18 sept.  2018  index.html
-rw-r--r-- 1 fred fred   7092 17 sept.  2018  index.html~
-rw-r-xr-x 1 fred root   2208 17 sept.  2018  LabStats
-r--r--r-- 1 fred fred    344 24 sept.  2018  myExtractor1.m
-r--rw-r-- 1 fred fred    344 26 sept.  2018  myExtractor2.m
-r--r--r-- 1 fred fred    427 27 sept.  2018  myExtractor3.m
----rw-r-- 1 fred fred    177 17 sept.  2018  myFisher.m
-rw-r--r-- 1 fred fred   1107 24 sept.  2018  ransac.m
-rw-r--r-- 1 fred fred   1277 24 sept.  2018  ransac.m~
drw-r-xr-- 2 fred info   4096 12 oct.   2018  Releves
-------r-- 1 fred info 242573 17 sept.  2018  toto.mat
$
```

Questions :

1. Sur quels fichiers fred n'a t'il pas le droit de lecture ? D'écriture ? D'exécution ?

Sol.:

Fichier	lecture interdite	écriture interdite	exécution interdite
`Calltech101`			
`cent_moment.m`			
`evalBOF`		X	X
`#evalBVF.m#`			X
`evalBVF`			X
`feature_vec.m`			X
`feature_vec`	X	X	
`feature_res`			X
`index.html`			X
`index.html~`			X
`LabStats`			X
`myExtractor1.m`		X	X
`myExtractor2.m`		X	X
`myExtractor3.m`		X	X
`myFisher.m`	X	X	X
`ransac.m`			X
`ransac.m~`			X
`Releves`			X
`toto.mat`	X	X	X

2. Quelles commandes devrait-il lancer pour que les utilisateurs du groupe primaire `info` puissent lire le fichier `toto.mat`, et qu'il puisse lui-même lire et écrire sur ce fichier ?

   Sol.: chmod g+r toto.mat
   chmod u+rw toto.mat
   En une seule commande : chmod g+r,u+rw toto.mat
   En octal : chmod 644 toto.mat

3. Supposez qu'à partir du répertoire listé, il lance la commande `ls␣-l␣Releves`. Cette dernière réussira t'elle ?

   Sol.: Non, car même s'il a acquis les droits de lire et d'écrire dans le répertoire, il a utilisé l'option -l de ls, qui nécessite l'accès aux informations des fichiers que contient le répertoire, et non du répertoire lui-même. Or, le x n'est pas positionné dans les droits U, ce qui lui interdit d'aller plus loin que le répertoire.

4. Le programme binaire `features_vec` a besoin d'ouvrir le fichier `features_res` en écriture pour y ajouter des résultats. Le pourra t'il s'il est lancé par fred ?

   Sol.: Non, car si `fred:fred` lance `features_vec`, alors ce dernier sera exécuté en tant que `eric:info` car c'est un fichier binaire d'après l'énoncé, et les bits SetUID et SetGUID sont tous les deux positionnés. Or :
   — eric $\neq fred \rightarrow$ ne peut pas acquérir les droits U
   — et l'énoncé ne dit pas que `eric` est du groupe `fred` $\rightarrow$ ne peut pas acquérir les droits G
   — il acquiert donc les droits O sur le fichier, qui sont vides.

2 Jouer avec les fichiers

2.1 La commande ls

Utilisée sans arguments, la commande ls vous permet de lister les fichiers du répertoire courant. Consultez sa page manuel ('q' pour la quitter) : man ls

1. Placez-vous dans votre répertoire par défaut, et appelez ls. Appelez ensuite ls avec les options -l et -l -a, soit ls -l et ls -la Quel est l'effet de -l ? De -a ?

Sol.: L'option -l a pour effet d'afficher les fichiers en format "long" (informations étendues), et -a affiche aussi ceux dont le nom commence par un "." (fichiers de configuration, normalement cachés).

2. Listez le contenu de la racine (/) avec les options -l et -a

   Sol.: ls -la /

3. A quoi pourrait correspondre le premier caractère 'd' précédent les droits ? Vérifier cela en appelant à nouveau ls avec le bon argument.

   Sol.: Il apparaît lorsque l'entrée est un répertoire. Par exemple, /usr et /tmp sont des répertoires et apparaissent comme tels.

2.2 La commande cp

La commande cp vous permet de copier des fichiers. Il y a deux formes possibles :
— cp src dest : copie un seul fichier source src vers le fichier cible dest
— cp src1 src2 ... dest : copie plusieurs fichiers src1, src2, ... vers le répertoire dest
Consulter sa page manuel : man cp

1. Copiez le fichier `/etc/passwd` vers votre répertoire par défaut. Avec `ls␣-l`, listez `/etc/passwd` et `~/passwd` (en une ou deux commandes, peu importe) La datation est-elle la même ?

   Sol.:

```
[xavier@localhost ~]$ cp /etc/passwd ~
[xavier@localhost ~]$ ls -l /etc/passwd ~/passwd
-rw-r--r-- 1 root   root   3079 17 déc.  12:48 /etc/passwd
-rw-r--r-- 1 xavier xavier 3079 16 avril 18:59 /home/xavier/passwd
[xavier@localhost ~]$
```

   Le fichier nouvellement créé apparaît daté avec le jour et l'heure de la copie, donc la datation d'origine est perdue.

2. Supprimez le fichier `~/passwd` en faisant `rm␣~/passwd` Puis refaites la copie, mais avec l'option -p de cp. Et maintenant ?

   Sol.:

```
[xavier@localhost ~]$ cp /etc/passwd ~
[xavier@localhost ~]$ ls -l /etc/passwd ~/passwd
-rw-r--r-- 1 root   root   3079 17 déc.  12:48 /etc/passwd
-rw-r--r-- 1 xavier xavier 3079 16 avril 18:59 /home/xavier/passwd
[xavier@localhost ~]$
```

   Maintenant, elle est conservée !

3. La commande stat (status) vous permet d'obtenir toutes sortes d'informations sur un ou des fichiers. Faites un `stat␣/etc/passwd␣~/passwd` Les datations sont-elles toutes identiques ?

   Sol.:

```
[xavier@localhost ~]$ stat /etc/passwd ~/passwd
  Fichier : /etc/passwd
   Taille : 3079       Blocs : 8          Blocs d'E/S : 4096   fichier
Périphérique : 10304h/66308d Inœud : 101476557   Liens : 1
Accès : (0644/-rw-r--r--)  UID : (    0/    root)   GID : (    0/    root)
 Accès : 2023-04-16 11:03:54.723988802 +0200
Modif. : 2022-12-17 12:48:05.483326647 +0100
Changt : 2022-12-17 12:48:05.486326508 +0100
  Créé : -
```

```
  Fichier : /home/xavier/passwd
   Taille : 3079       Blocs : 8          Blocs d'E/S : 4096   fichier
Périphérique : 10303h/66307d Inœud : 10752531    Liens : 1
Accès : (0644/-rw-r--r--)  UID : ( 1000/  xavier)   GID : ( 1000/  xavier)
 Accès : 2023-04-16 11:03:54.723988802 +0200
Modif. : 2022-12-17 12:48:05.483326647 +0100
Changt : 2023-04-16 19:01:34.993275556 +0200
  Créé : -
[xavier@localhost ~]$
```

On voit qu'il y a en fait 3 datations, qui renseignent sur les dates de dernier accès, de dernière modification (du contenu) et de dernier changement (de status, comme les droits). Même si la date de modification a été préservée par -p, on voit que le fichier a été modifié après avoir été accédé.

2.3 La commande ln

La commande ln (link) vous permet de créer un lien symbolique (avec l'option -s) ou matériel (sans l'option -s).

1. Dans le terminal, refaites un ls -l Puis lancez `ln␣~/passwd␣~/lien`, et refaites encore un ls -l Qu'est-ce qui a changé ? Vérifiez avec `stat`.

   Sol.:

```
[xavier@localhost ~]$ ls -l ~/passwd
-rw-r--r-- 1 xavier xavier 3079 17 déc.  12:48 /home/xavier/passwd
[xavier@localhost ~]$ ln ~/passwd ~/lien
[xavier@localhost ~]$ ls -l ~/passwd ~/lien
-rw-r--r-- 2 xavier xavier 3079 17 déc.  12:48 /home/xavier/lien
-rw-r--r-- 2 xavier xavier 3079 17 déc.  12:48 /home/xavier/passwd
[xavier@localhost ~]$
```

   Le '1' qui figurait après les droits, est maintenant passé à '2'. Ce même '2' est confirmé par stat dans le champs "Liens" :

```
[xavier@localhost ~]$ stat ~/passwd ~/lien
  Fichier : /home/xavier/passwd
   Taille : 3079       Blocs : 8          Blocs d'E/S : 4096   fichier
Périphérique : 10303h/66307d Inœud : 10752531    Liens : 2
Accès : (0644/-rw-r--r--)  UID : ( 1000/  xavier)   GID : ( 1000/  xavier)
 Accès : 2023-04-16 11:03:54.723988802 +0200
Modif. : 2022-12-17 12:48:05.483326647 +0100
Changt : 2023-04-16 19:16:43.774004697 +0200
  Créé : -
  Fichier : /home/xavier/lien
   Taille : 3079       Blocs : 8          Blocs d'E/S : 4096   fichier
Périphérique : 10303h/66307d Inœud : 10752531    Liens : 2
Accès : (0644/-rw-r--r--)  UID : ( 1000/  xavier)   GID : ( 1000/  xavier)
 Accès : 2023-04-16 11:03:54.723988802 +0200
Modif. : 2022-12-17 12:48:05.483326647 +0100
Changt : 2023-04-16 19:16:43.774004697 +0200
  Créé : -
[xavier@localhost ~]$
```

2. Gedit (Gnome editor) est un éditeur texte. Editez votre fichier passwd en faisant `gedit␣~/passwd` Supprimez une ou deux lignes, puis enregistrez-le. Retournez dans le

shell, et listez à nouveau `~/passwd` et `~/lien` Conclusion ? (Si vous avez un doute : la commande diff fic1 fic2 compare les fichiers fic1 et fic2 et affiche les différences quand elle en trouve. Comparez donc `~/passwd` et `/etc/passwd`, puis `~/passwd` et `~/lien`)

3 Résolution des noms de fichiers

Cet exercice doit être fait sur machine

1. Ouvrez un terminal, et essayez-y les commandes suivantes :

```
cd
echo $PWD
echo ~
cd /tmp
echo $PWD
echo ~
```

   Quels résultats du cours retrouvez-vous ici ?

2. En utilisant ls ou echo, ou mieux encore, le programme `args1.c` vu en cours, faire afficher les noms de fichiers ou de répertoires :
   (a) de 3 lettres exactement, situés au niveau 0 (ç-à-d rattachés directement à la racine)

   Sol.: echo /???

   (b) qui se terminent par m, au niveau 1

   Sol.: echo /*/*m

   (c) qui se terminent par un chiffre, au niveau 1

   Sol.: echo /*/*[0-9]

   (d) d'au moins 3 lettres, figurant dans /etc, qui contiennent un caractère 'e' mais pas parmi ses 3 premières lettres

   Sol.: `echo␣/etc/[^e][^e][^e]*e*`

   (e) figurant dans /etc, et dont l'une des trois dernières lettres est un caractère 'e'

   Sol.: echo /etc/*e /etc/*e? /etc/*e??

   (f) qui contiennent au moins un chiffre, au niveau 1

   Sol.: echo /*/*[0-9]*

   (g) qui sont dans /etc ou ses descendants immédiats, et ne débutent pas par une lettre majuscule

   Sol.: `echo␣/etc/[^A-Z]*␣/etc/*/[^A-Z]*`

4 Redirections

4.1 La commande find

Forme générale : `find␣répertoires␣prédicats`

Elle parcourt récursivement l'aborescence à partir d'un ou plusieurs répertoires spécifiés et affiche les fichiers qui concordent avec le ou les `prédicats`.

Par exemple `find␣/etc␣-type␣d` listera les entrées de type d = directoy = répertoire à partir de `/etc`.

1. Modifier la ligne précédente pour que la sortie standard de find termine dans un fichier `/tmp/sortie`. Vérifier que c'est bien correct en faisant cat `/tmp/sortie`.

   Sol.:

```
find /etc -type d > /tmp/sortie
cat /tmp/sortie
```

2. Compléter la ligne de commande précédente pour que la sortie d'erreur standard termine dans un fichier `/tmp/erreurs`. Vérifier le fichier produit.

   Sol.:

   ```
   find /etc -type d > /tmp/sortie 2> /tmp/erreurs
   cat /tmp/erreurs
   ```

3. Modifier la ligne précédente pour que les deux sorties standard et d'erreur standard terminent un même fichier `/tmp/tout`. Vérifier le fichier produit là encore.

   Sol.:

   ```
   find /etc -type d > /tmp/tout 2>&1
   cat /tmp/tout
   ```

4.2 La commande wc

La commande `wc` compte combien de lignes, mots, et caractères contient son entrée standard. Elle ne peut terminer que lorsque l'entrée standard ne peut plus être alimentée, autrement dit, lorsqu'elle est fermée ; ce qui se fait par la combinaison de touches Ctrl+D lorsqu'elle est rattachée à un terminal.

1. Commencez par tester wc seule dans un terminal, en tapant quelques lignes/mots.

   Sol.:

   ```
   $ wc
   blabla blaabl
   un mot
   deux mots
   des mots
   ^D
         4       8      40
   $
   ```

2. Faites maintenant en sorte que `wc` opère sur le fichier `/tmp/sortie` que vous avez produit à l'exercice précédent.

   Sol.:

   ```
   $ wc < /tmp/sortie
     517   517 12567
   $
   ```

4.3 La commande cat

La commande cat concatène le contenu du ou des fichiers passés en arguments vers sa sortie standard.

1. Essayez cat `/etc/passwd`, puis `cat␣/etc/group`, et enfin `cat␣/etc/passwd␣/etc/group` pour vous en convaincre.
2. Concaténer les fichiers `/tmp/sortie` et `/tmp/erreurs` en un seul fichier `/tmp/toto`.

   Sol.:

```
$ find /etc/ -type d > /tmp/tout 2>&1
$ lscat /tmp/sortie /tmp/
$ ls -l /tmp/sortie /tmp/erreurs
-rw-r--r-- 1 hilairex hilairex  1321 14 mars  15:09 /tmp/erreurs
-rw-r--r-- 1 hilairex hilairex 12567 14 mars  15:09 /tmp/sortie
$ cat /tmp/sortie /tmp/erreurs > /tmp/toto
$ ls -l /tmp/toto /tmp/tout
-rw-r--r-- 1 hilairex hilairex 13888 14 mars  15:10 /tmp/toto
-rw-r--r-- 1 hilairex hilairex 13888 14 mars  15:09 /tmp/tout
$
```

Les tailles des fichiers `/tmp/tout` et `/tmp/toto` sont-elles différentes ? Et leur contenu ?

3. Refaites la question 2 de l'exercice 4.2 en utilisant cat et une autre méthode de redirection.

   Sol.:

```
$ cat /tmp/sortie | wc
    517     517   12567
$
```

4. Reproduire à nouveau le même résultat, mais sans utiliser de fichier /tmp/sortie

   Sol.:

```
$ find /etc/ -type d 2>/dev/null | wc
    517     517   12567
$
```

   NOTE : /dev/null est un fichier spécial toujours vide – c'est une sorte de « poubelle » 100% écologique : elle est vide en permanence, et à coût de stockage nul.

5 Variables ordinaires et d'environnement

5.1 Variable ordinaire

Supposez que `fic` contienne un nom de fichier avec chemin absolu – pour le seul besoin de l'exemple, disons `/etc/libssh/libssh_client.config`

En utilisant echo, faites afficher :

1. Le chemin seul de fic, terminé par un '/' (`/etc/libssh/` dans l'exemple). Cela fonctionne t'il toujours avec un fichier situé à la racine ?

   Sol.:

```
$ fic=/etc/libssh/libssh_client.config
$ echo ${fic%/*}
/etc/libssh
$
```

2. Le nom du fichier seul (`libssh_client.config` dans l'exemple)

   Sol.:

```
$ echo ${fic##*/}
libssh_client.config
$
```

3. L'extension du fichier seule (`.config` dans l'exemple).

   Sol.:

```
$ echo ${fic##*.}
config
$
```

Obtenez-vous bien une extension vide si le fichier n'en a pas ?

Sol.:

```
$ fic=/machin/bidule/chose
$ echo ${fic##*.}
/machin/bidule/chose
$
```

Malheureusement, non. Si un fichier n'a pas d'extension, alors il n'y a pas de caractère '.' donc la confrontation de son nom à '*.' échoue, ce qui laisse la chaîne d'origine inchangée.

Une solution possible pour contourner ce problème consiste à opérer en 2 temps : i) on commence par retirer l'extension du fichier en rabotant par la droite, ii) on utilise ensuite la chaîne résultat pour raboter par la gauche le nom du fichier d'origine

```
$ f=/machin/bidule/chose.ext
$ echo ${f%.*}
/machin/bidule/chose
$ g=${f%.*}
$ echo ${f#$g}
.ext
$ f=/machin/bidule/chose
$ echo ${f%.*}
/machin/bidule/chose
$  g=${f%.*}
$  echo ${f#$g}

$
```

5.2 Variable d'environnement PATH

Dans le cours, nous avons utilisé deux programmes `args1` et `args2`, que nous avons toujours lancés à partir du répertoire courant, ce qui nécessite de les faire précéder soit par le chemin absolu, soit par ./

1. Créer un répertoire bin rattaché directement à votre répertoire par défaut (HOME)

   Sol.:

   ```
   mkdir ~/bin
   ```

2. Placez-vous dans bin, téléchargez les deux programmes `args1.c` et `args2.c` et recompilez-les, comme cela est fait dans le cours

   Sol.:

   ```
   $ cd ~/bin
   $ wget https://perso.esiee.fr/~hilairex/3R-IN3/args1.c 2> /dev/null
   $ wget https://perso.esiee.fr/~hilairex/3R-IN3/args2.c 2> /dev/null
   $ ls -l args?.c
   -rw-r--r--. 1 hilairex hilairex 207 17 avril  2023 args1.c
   -rw-r--r--. 1 hilairex hilairex 300 17 avril  2023 args2.c
   $ gcc -o args1 args1.c
   $ gcc -o args2 args2.c
   $
   ```

3. Affichez le contenu de votre variable PATH. Que devez-vous lui ajouter pour que le système aille aussi examiner votre répertoire bin après tous les autres ?

   Sol.:

```
$ PATH="$PATH:$HOME/bin"
$ echo $PATH
/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/home/hilairex/bin
$
```

4. Vérifiez que votre solution est bien opérationnelle : on doit pouvoir lancer `args1` et `args2` depuis n'importe quel répertoire, sans avoir à préciser de chemin.

   Sol.:

```
$ args1 un 2 trois
Ici args1, qui a reçu 4 arguments:
argv[0]=args1
argv[1]=un
argv[2]=2
argv[3]=trois
$ args2 un deux 3 quatre
Ici args2, qui a reçu 5 arguments:
argv[0]=args2
argv[1]=un
argv[2]=deux
argv[3]=3
argv[4]=quatre
envp[0]=IMSETTINGS_INTEGRATE_DESKTOP=yes
envp[1]=SHELL=/bin/bash
envp[2]=SESSION_MANAGER=local/unix:@/tmp/.ICE-unix/32439,unix/unix:/tmp/.ICE-unix/3243
envp[3]=COLORTERM=truecolor
```

   (sortie coupée)

5. Exécutez une ligne de commande qui ajoute la ligne précédente à votre fichier `~/.bashrc` En ouvrant un nouveau terminal, vous devriez obtenir la variable PATH étendue, et non celle d'origine.

   Sol.:

```
$ echo 'PATH=$PATH:$HOME/bin' >> ~/.bashrc
$ source ~/.bashrc
$ echo $PATH
/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/home/hilairex/bin
$
```