Système de Gestion de fichiers sous Linux

TP1 : Commandes Linux

Application à la programmation en C – Appels systèmes et librairie standard

Savoirs et Compétences :

S4 - Développement Logiciel

1 – Principes de base : Flux d'entrée et de sortie de base : terminaux, fichiers, … (spécifications POSIX)
3 - Structure et gestion des données : Formats de fichier : texte (human readable), binaire …

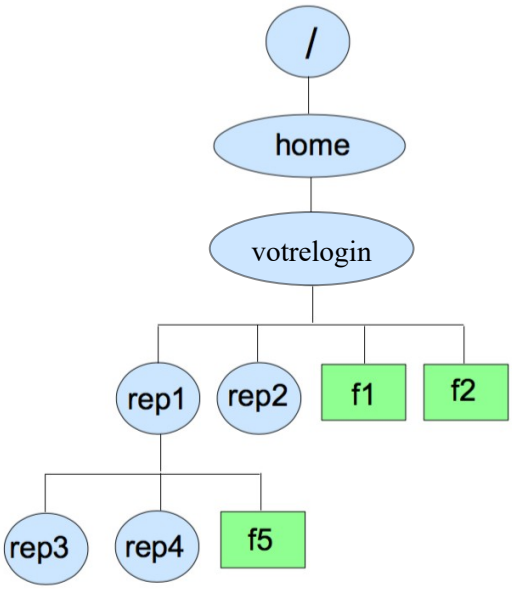
S6 – Systèmes d'exploitation

1 - Notions fondamentales : Système de droits des utilisateurs
2 - Machines virtuelles : Installation / configuration

Etape 1 : Les droits d'accès aux fichiers et aux répertoires Les inoeuds, les liens directs, les liens symboliques

Sous LINUX, la plupart des commandes sont accessibles via l'interface graphique ou via le clavier (dans une fenêtre appelée « Terminal »).
L'énoncé de cette étape insiste sur les commandes du Terminal, mais vous devrez également constater chaque fois que possible que l'interface graphique permet de réaliser les mêmes opérations de façon plus conviviale.

*Sauf indication contraire, vous devez vous connecter en tant qu'utilisateur de l'ordinateur (login « **votrelogin** » mot de passe « **votremotdepasse** ») et non pas en tant qu'administrateur admini ou administrateur root.*
*Si vous êtes admin au moment de la connexion (la commande **whoami** vous permet de le savoir), déconnectez-vous et **reconnectez-vous en tant qu'utilisateur de la machine**.*



Système de Gestion de fichiers sous Linux

TP1 : Commandes Linux

Application à la programmation en C – Appels systèmes et librairie standard

1.1 Créer dans votre répertoire privé, la hiérarchie de répertoires donnée ci-dessus

Les droits d'accès aux fichiers et répertoires (commandes chmod et umask)

1.2 Dans le répertoire maison, créer un fichier texte nommé **test1**, en utilisant un éditeur de texte (**gedit ou Sublime-text à installer avec la commande aptitude update ; aptitude install gedit ou apt-get update ; apt-get install gedit**).

1.3 Écrire les caractères “abcd” à l’intérieur, sauvegarder dans le fichier puis fermer **gedit**.

1.4 Utiliser la commande :

```
$ chmod 762 test1
```

1.5 Vérifier les droits du fichier **test1**. Expliquer la démarche utilisée et les résultats obtenus.

Pour voire les droits on utilise la commande :

```
$ ls -l test1
```

Résultat obtenue :

```
ejehanno@snir-VirtualBox:~$ ls -l test1
-rwxrw--w- 1 ejehanno ejehanno 5 sept. 22 16:13 test1
```

1.6 Initialiser le masque des droits de création des fichiers de façon à ce que, pour les fichiers créés, les droits du propriétaire ne soient pas modifiés lors des copies et que les droits du groupe propriétaire et des autres utilisateurs soient automatiquement supprimés. Quelle commande avez-vous entrée ?

```
$ umask 077
```

Système de Gestion de fichiers sous Linux

TP1 : Commandes Linux

Application à la programmation en C – Appels systèmes et librairie standard

1.7 Copier le fichier test1 dans les fichiers **test2** et **test3** se trouvant dans les répertoires **rep1**/**rep3** de 2 façons différentes (commande **cp** et **cat**). Vérifier les droits des fichiers **test2** et **test3** en utilisant la commande **ls -l**. Donnez les commandes entrées et expliquez les résultats obtenus.

Résultat avec la commande « cp » :

```
ejehanno@snir-VirtualBox:~/rep_privée$ cp test1 ./rep1/test2 | cp test1 ./rep1/rep3/test3
ejehanno@snir-VirtualBox:~/rep_privée$ tree
.
├── f1
├── f2
├── rep1
│   ├── f5
│   ├── rep3
│   │   └── test3
│   ├── rep4
│   └── test2
├── rep2
└── test1
```

```
ejehanno@snir-VirtualBox:~/rep_privée$ ls -l ./rep1/test2
-rwx------ 1 ejehanno ejehanno 5 sept. 22 16:35 ./rep1/test2
ejehanno@snir-VirtualBox:~/rep_privée$ ls -l ./rep1/rep3/test3
-rwx------ 1 ejehanno ejehanno 5 sept. 22 16:35 ./rep1/rep3/test3
```

Résultat avec la commande « cat » :

```
ejehanno@snir-VirtualBox:~/rep_privée$ cat test1 > ./rep1/test2 | cat test1 > ./rep1/rep3/test3
ejehanno@snir-VirtualBox:~/rep_privée$ tree
.
├── f1
├── f2
├── rep1
│   ├── f5
│   ├── rep3
│   │   └── test3
│   ├── rep4
│   └── test2
├── rep2
└── test1
```

```
ejehanno@snir-VirtualBox:~/rep_privée$ ls -l ./rep1/test2
-rw------- 1 ejehanno ejehanno 5 sept. 22 16:42 ./rep1/test2
ejehanno@snir-VirtualBox:~/rep_privée$ ls -l ./rep1/rep3/test3
-rw------- 1 ejehanno ejehanno 5 sept. 22 16:42 ./rep1/rep3/test3
```

La commande cat enlevé les droits d'exécution.

Système de Gestion de fichiers sous Linux

TP1 : Commandes Linux

Application à la programmation en C – Appels systèmes et librairie standard

1.8 Recommencer l'opération de copie de ***test1*** vers ***test4*** de manière à ce que ***test4*** ait ***nativement les droits rw-rw-r--***. Quelle commande permet une recréation des droits ? Pourquoi ?

```
ejehanno@snir-VirtualBox:~/rep_privée$ install -m 664 test1 test4
ejehanno@snir-VirtualBox:~/rep_privée$ ls -l
total 16
-rw-rw-r-- 1 ejehanno ejehanno    0 sept. 22 15:34 f1
-rw-rw-r-- 1 ejehanno ejehanno    0 sept. 22 15:34 f2
drwxrwxr-x 4 ejehanno ejehanno 4096 sept. 22 16:42 rep1
drwxrwxr-x 2 ejehanno ejehanno 4096 sept. 22 15:34 rep2
-rwxrw--w- 1 ejehanno ejehanno    5 sept. 22 16:13 test1
-rw-rw-r-- 1 ejehanno ejehanno    5 sept. 22 17:06 test4
```

1.9 Quels sont les droits d'accès à votre répertoire privé ? Permettent-ils une sécurisation suffisante ?

```
ejehanno@snir-VirtualBox:/home$ ls -l
total 8
drwxr-xr-x 16 ejehanno ejehanno 4096 sept. 22 17:05 ejehanno
```

Système de Gestion de fichiers sous Linux

TP1 : Commandes Linux

Application à la programmation en C – Appels systèmes et librairie standard

Rôles des bits s sur les fichiers

*A l'instar de la commande **userdel** qui travaille sur le fichier original contenant les utilisateurs du système linux, le fichier **suppasswd** est conçu pour supprimer dans le fichier **copie_passwd** l'entrée correspondante à un utilisateur.*

1.10 Copier le fichier **suppasswd** dans votre répertoire maison et en faire la propriété de l'utilisateur **root** et du groupe **root** de manière à obtenir le résultat ci-dessous :

-rw-r--r-- 1 root root 2534 sept. 21 16:35 copie_passwd

Décrivez les commandes utilisées pour réaliser cette opération.

```
ejehanno@snir-VirtualBox:~/rep_privée$ cp suppasswd copie_passwd
```

La commande chown permet de modifier le propriétaire du fichier ainsi que le groupe.

```
snir@snir-VirtualBox:/home/ejehanno/rep_privée$ sudo chown root:root copie_passwd
ejehanno@snir-VirtualBox:~/rep_privée$ ls -l copie_passwd
-rw-rw-r-- 1 root root 9080 sept. 22 17:28 copie_passwd
```

1.11 Copier le fichier **/etc/passwd** dans **/home/votrecompte/copie_passwd**. Quel est le rôle de ce fichier et quelles informations contient-il ?

Le fichier /etc/passwd contient toutes les informations relatives aux utilisateurs (login, mots de passe, ...).

1.12 A l'aide des commandes **chown**, **chgrp** et **chmod** rendre les propriétés du fichier **copie*_passwd*** identiques à celle du fichier ***passwd*** d'origine.

```
ejehanno@snir-VirtualBox:/etc$ ls -l passwd
-rw-r--r-- 1 root root 2838 sept. 22 15:30 passwd
ejehanno@snir-VirtualBox:~/rep_privée$ ls -l copie_passwd
-rw-r--r-- 1 root root 2838 sept. 22 17:38 copie_passwd
```

Système de Gestion de fichiers sous Linux

TP1 : Commandes Linux

Application à la programmation en C – Appels systèmes et librairie standard

1.13 Après lui avoir donné les droits nécessaires, exécuter le programme ***suppasswd*** en tant qu'utilisateur de l'ordinateur. Que constate-t-on ?

Le programme se lance normalement .

1.14 Modifier les droits du fichier ***suppasswd*** de manière à positionner **s** sur la première série des droits (u) du fichier. Relancer l'exécution de ***suppasswd***. Conclusion.

```
ejehanno@snir-VirtualBox:~/rep_privée$ chmod u+s suppasswd
ejehanno@snir-VirtualBox:~/rep_privée$ ls -l suppasswd
-rwsrw-r-- 1 ejehanno ejehanno 9080 sept. 22 17:26 suppasswd
ejehanno@snir-VirtualBox:~/rep_privée$ ./suppasswd
Quel utilisateur voulez-vous supprimer ? ->
```

Le programme s'exécute mais il y a une erreur à cause du droit « s » qui donne les droit du propriétaire du fichier .

```
ejehanno@snir-VirtualBox:~/rep_privée$ ./suppasswd
Quel utilisateur voulez-vous supprimer ? -> erwan
Erreur de segmentation (core dumped)
```

Nombre de liens directs sur fichiers et sur répertoires

*Vous êtes l'utilisateur **invite** du groupe **invite** et le masque des droits de création des fichiers vaut 077.*

1.15 Créer un fichier **essai** et lui donner les droits d'accès **–rwx rwx rwx**

```
ejehanno@snir-VirtualBox:~/rep_privée$ chmod 777 essai
ejehanno@snir-VirtualBox:~/rep_privée$ ls -l essai
-rwxrwxrwx 1 ejehanno ejehanno 0 sept. 23 08:24 essai
```

Système de Gestion de fichiers sous Linux

TP1 : Commandes Linux

Application à la programmation en C – Appels systèmes et librairie standard

1.16 Frapper les commandes :

```
$ ls -il
$ cp essai essai1
$ mv essai essai2
$ ln essai2 essai3
$ ls -il
```

Expliquer pourquoi les droits des fichiers ***essai1*** et ***essai2*** sont différents.

Expliquer pourquoi les **inoeuds** et les droits d'accès des fichiers ***essai2*** et ***essai3*** sont identiques. Où est passé le fichier ***essai*** ?

Le fichier essai est devenu essai2 et les inoeuds et les droit sont identiques car il sont des liens de l'ancien essai.

1.17 Frapper la commande :

$ **ls -il** /

Combien de répertoires contient le répertoire */home* ?

Il y a 2 répertoire :

```
drwxr-xr-x 17 ejehanno ejehanno 4096 sept. 23 08:09 ejehanno
drwxr-xr-x 28 snir     snir     4096 sept. 22 15:27 snir
```

Combien vaut son nombre de liens directs ? Pourquoi ?

```
917505 drwxr-xr-x   4 root root      4096 sept. 22 15:29 home
```

Il y a 4 lien direct : «ejehanno» , « snir » , « . » et « .. »

Système de Gestion de fichiers sous Linux

TP1 : Commandes Linux

Application à la programmation en C – Appels systèmes et librairie standard

1.18 Passer en utilisateur **root**.

1.19 Connecter une clé USB. A l'aide de la commande **mount** (voir le man), monter le système de fichiers de la clé vers le répertoire **rep2**.
Tester alors la création, dans le compte **/home/admin** d'un lien direct puis d'un lien symbolique vers un fichier de la clé. Conclusion.

```
Les liens physiques ne marcheent pas car ils redirigent  directement
     sur le contenu .

Les liens symboliques est un lien qui redirige vers un fichier et pas
     directement le contenu .
```

Système de Gestion de fichiers sous Linux

TP1 : Commandes Linux

Application à la programmation en C – Appels systèmes et librairie standard

Etape 2 : Manipulation des fichiers en c/c++ Etude de quelques appels systèmes Différences entre les appels systèmes et les fonctions d'E/S des bibliothèques standards

*Tout OS fourni des points d'entrée au noyau sous forme **d'interaction de niveau 1, 2 ou 3** (cf cours 1). Plus le niveau de l'interaction est bas plus l'interaction est forte. Les interactions de niveau 1 sont fréquemment appelées « **appels systèmes** » ou « **primitives systèmes** ». Dans cette partie de TP nous allons observer ce qui distingue les « appels systèmes » des fonctions des bibliothèques standards en ce qui concerne l'accès et la gestion des fichiers.*
*En ce qui concerne les « primitives systèmes » sous linux, on utilisera les normes C-POSIX fournies par les bibliothèques **unistd.h, fcntl.h, sys/stat.h, sys/types.h**.*
*Pour ce qui concerne les fonctions standards on utilisera la bibliothèque **stdio.h***

*Le compilateur utilisé sera **gcc** ou **g++** selon que l'on utilisera c ou c++ comme langage de programmation.*

*La syntaxe d'utilisation de la commande gcc est la suivante : **gcc fichier1.c –o prog1***

*Le manuel linux (man) fournira les aides sur l'utilisation des fonctions et primitives systèmes. Utiliser pour cela la syntaxe "**man n°section fonction**" avec n°section = section du manuel et fonction = fonction à utiliser en c/c++ (cf man man).*

TU01 :

2.1 Utiliser la bibliothèque standard du langage C pour écrire un caractère dans un fichier « **ff** » (fonctions **fopen** et **fwrite**).
Le programme **TU01** se terminera par une boucle infinie et devra être interrompu par **^c (ctrl c)**.
Lister le contenu du fichier "ff". En vous référant au cours, expliquer pourquoi le caractère n'est pas dans le fichier ?

Le caractère n'est pas dans le fichier « ff » car il n'a était enregistré parce que la mémoire temp ne c'est pas vidé.

Système de Gestion de fichiers sous Linux

TP1 : Commandes Linux

Application à la programmation en C – Appels systèmes et librairie standard

TU02 :

2.2 Utiliser les appels systèmes d'E/S **open** et **write** pour écrire un caractère dans le fichier de référence ff. le programme **TU02** se terminera par une boucle infinie et devra être interrompu par un **^c**.
Lister le contenu du fichier ff. Pourquoi le caractère est-il dans le fichier ?

TU03 :

2.3 Reprendre le programme TU01 et faire précéder la boucle infinie par la fonction **fflush**().
Conclusion.

Fflush() vide la mémoire tampon avant de continuer, donc les caractères sont dans le fichier.

3.3 Reprendre le programme TU01 et remplacer la boucle infinie par la fonction **fclose()** .
Conclusion.

Fclose() ferme le fichier avec un fflush() donc le caractère et aussi dans le fichier.

3.4 Reprendre le programme TU01 et remplacer la fonction **fclose()** par la fonction **exit()**.
Conclusion.

La mémoire tampon est vidé et le programme se ferme.

3.5 Reprendre le programme TU01 et utiliser la fonction **setbuf()** avec un nouveau tampon nommé **newtamp** puis avec la valeur **NULL.** Conclusion.

Si buffer = NULL on indique au buffer qu'il ne doit pas être utilisé donc tous les caractère écrit dans le fichier vons être écrit sans passer par le buffer.

Système de Gestion de fichiers sous Linux

TP1 : Commandes Linux

Application à la programmation en C – Appels systèmes et librairie standard

TU04 :

3.6 Reprendre les questions 3.1 à 3.6 en remplaçant le fichier ff par la **sortie standard**. Conclusion.

Les caractère s'affiche sur l'écran car la sorti standard est l'écran.

Redirection de la sortie standard vers un fichier

Système de Gestion de fichiers sous Linux

TP1 : Commandes Linux

Application à la programmation en C – Appels systèmes et librairie standard

TU05 :

3.7 Utiliser les appels systèmes **open** – en mode O_WRONLY|O_CREAT|O_EXCL - **close** et **dup** pour rediriger la sortie standard dans un fichier de référence **ff**. Faire un schéma des différentes tables et pointeurs concernés :

- Après le lancement du programme
- Après appel à open
- Après appel à close
- Après appel à dup

Expliquer.

3.8 Après les appels systèmes de redirection, ajouter un appel à la fonction printf() de la bibliothèque C.
Conclusion

3.9 Après les appels systemes de redirection, ajouter un appel à **std ::cout** de la bibliothèque C++.
Conclusion.

3.10 Comment peut-on effectuer la redirection sans effacer (avec la commande **rm**) le fichier **ff**, mais en rajoutant les caractères en fin de fichier ?

Système de Gestion de fichiers sous Linux

TP1 : Commandes Linux

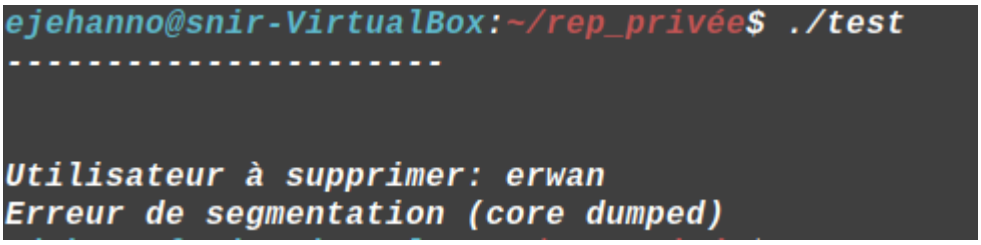
Application à la programmation en C – Appels systèmes et librairie standard

Exercice d'application 1 :

3.11 Réaliser l'implémentation du programme **suppasswd** précédemment utilisé.

*Pour ce faire, on pourra ouvrir 2 fois le fichier **copie_passwd** : une première fois pour le lire entièrement, une seconde pour le réécrire.*
*On utilisera la bibliothèque standard du C et on ouvrira le fichier **copie_passwd** en mode **troncature** pour la réécriture du fichier.*
On pourra aussi utiliser un fichier temporaire qu'on effacera ensuite si le besoin s'en fait sentir.

3.12 Refaire la question 2.12 en **c++** et en utilisant la bibliothèque **fstream**



Système de Gestion de fichiers sous Linux

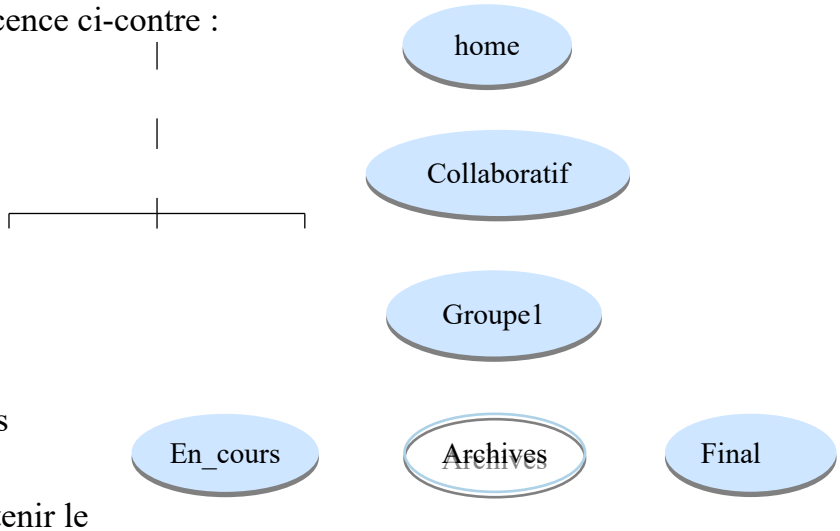
TP1 : Commandes Linux

Application à la programmation en C – Appels systèmes et librairie standard

Exercice d'application 2 (facultatif) :

*En reprenant l'idée de l'exercice 4 du **TD – Rappels Linux**, on souhaite créer une structure collaborative de travail. Dans cette structure, tous les membres du groupe **gpe1** pourront créer, supprimer, lire et modifier leurs propres fichiers dans le répertoire **Groupe1**. Les autres membres du groupe auront la possibilité de lire et modifier les fichiers mis en partage alors que les autres utilisateurs n'auront aucun accès à la structure collaborative. Une fois le travail terminé le ou les fichier(s) seront déplacés vers le répertoire **Final** où, dès lors, aucune suppression ou modification ne sera possible. Un script simple permettra, en outre, de conserver une copie des fichiers dans un répertoire caché nommé **Archives**.*

3.13 Créer l'arborescence ci-contre :



3.14 Configurer les champs propriétaire et groupe propriétaire afin d'obtenir le fonctionnement souhaité.

3.15 Configurer les droits des répertoires ainsi que le **umask** afin d'obtenir le fonctionnement souhaité.

3.16 Vérifier que le fonctionnement souhaité sur la création et l'utilisation des fichiers est bien obtenu. Expliquer la démarche.

Système de Gestion de fichiers sous Linux

TP1 : Commandes Linux

Application à la programmation en C – Appels systèmes et librairie standard

3.17 Pour compléter les éléments de la structure collaborative de la page 3, réfléchir à un principe permettant de conserver les données des fichiers de travail à tout instant même en cas de suppression accidentelle d'un de ces fichiers. La solution proposée devra minimiser l'espace disque.

3.18 A l'aide des commandes linux (manuellement) mettre en œuvre la solution.

3.19 Créer un programme C permettant d'automatiser l'étape précédente.

3.20 Faire en sorte que ce programme s'exécute à intervalle régulier. Expliquer la démarche.