Le terminal

Manipulations basiques

La majorité des interactions avec les systèmes Linux se font à travers des terminaux. Ce premier exercice a pour objectif de vous familiariser avec le terminal, et les capacités de celui-ci.

1. Ouvrir un terminal
2. Faire afficher « bonjour ! » au terminal
3. Lister les fichiers du répertoire courant
4. Créer un répertoire « *test_dir* »
5. En une commande unique, créer un fichier nommé « *test_file* » dans le répertoire précédemment créé
6. Lister les fichiers contenus dans le répertoire « *test_dir* »
7. Se positionner dans le répertoire « *test_dir* »
8. Modifier le fichier « *test_file* » afin d'y ajouter du texte à votre discrétion
9. Afficher dans le terminal le contenu du fichier « *test_file* »
10. Renommer « *test_file* » en « *test_file_v2* »
11. Copier « *test_file_v2* » dans un fichier « *test_file_v2_2* »
12. Supprimer « *test_file_v2* »
13. Revenir dans le répertoire parent
14. Supprimer le répertoire « *test_dir* »

Scripts bash

Linux permet la création de scripts bash (fichiers en *.sh*). Ceux-ci permettent l'automatisation de différentes tâches et activités. Amusons-nous par exemple à automatiser l'exercice précédent.

L'objectif ici est de pouvoir, avec une commande simple, rejouer le premier exercice de ce TD.

1. Ouvrir un terminal
2. Créer un répertoire bac à sable « *bac_a_sable* »
3. Se positionner dans le répertoire « *bac_a_sable* »
4. Créer un fichier « *test_script.sh* »
5. Dans le fichier « *test_script.sh* » ajouter la ligne :

```
#!/bin/bash
```

Celle-ci donne un contexte au script, elle permet de définir ce qui va exécuter le code. Ici le programme « *bash* » donc

6. Sous cette ligne, ajouter :
    - Une ligne de commande permettant la création d'un répertoire « *test_dir* »
    - Une ligne de commande permettant, en une seule commande, la création et l'écriture de « toto » dans un fichier « *test_file.txt* »
    - Une ligne de commande permettant d'afficher dans le terminal le contenu de « *test_file.txt* »
    - Une ligne supprimant le répertoire « *test_dir* » et son contenu
    - Faire afficher « *0* » au terminal

Nota : Il peut s'avérer utile de tester vos commandes avant de les ajouter à votre script…

7. Exécuter le script à l'aide de la commande :

```
./ test_script.sh
```

8. L'exécution du script se solde par un échec :

```
./test_script.sh
-bash: ./test_script.sh: Permission denied
```

Par défaut, un script bash est créé sans les droits d'exécution. Modifier les droits liés à ce script afin de le rendre exécutable par n'importe qui

9. Lancer le script « *test_script.sh* » :

```
$ ./test_script.sh
toto
0
```

10. Pousser plus loin les capacités du script :
    - Permettez de donner du texte en argument au script. Celui-ci écrira ce texte dans le fichier « *test_file.txt* ».
    - Ajouter une condition au script : si on n'a pas donnée de texte en argument du script, le texte est « toto » par défaut et le script retourne « -1 » au lieu de « 0 »

Annexe A : Commandes et redirection

Les commandes Linux acceptent de nombreuses options dont on peut consulter la documentation en tapant *man ls* par exemple pour la commande *ls*. Celle-ci comporte des options pour affiche les fichiers cachés *ls -a* ou encore pour afficher les détails et permissions d'un fichier *ls -l*.

Commande	Description
`cd`	*Change Directory* Se déplacer dans l'arborescence
`ls`	*Lister* le contenu du répertoire courant
`mkdir`	Permet de créer un répertoire ***(make dir)***
`pwd`	*Affiche le dossier courant* (Print Workgin Directory)
`cp`	*Copier* des fichiers ou des répertoires
`mv`	*Déplacer (move)* ou renommer des fichiers ou des répertoires
`rm`	*Effacer (remove)* des fichiers ou des répertoires
`cat`	*Visualiser (concaténer)* le contenu d'un fichier
`echo`	*Afficher* un message ou le contenu d'une variable
`touch`	*Créer* un fichier vide ou réinitialiser le *timestamp* d'un fichier
`chmod`	Modifier les autorisations d'accès à un fichier
`nano`	*Éditeur de texte* en ligne de commande. Il y en a beaucoup
`ps`	Afficher les informations des *processus* en cours
`top`	*Gestionnaire de ressource* en TUI
`kill`	Envoyer un *signal* au processus, généralement pour l'arrêter
`grep`	*Filtrer* une sortie en ne gardant que les lignes contenant un terme

Linux permet la redirection des sorties des commandes.

Une commande *echo 'bonjour'* aura pour sortie « *bonjour* ». Cette sortie peut être redirigée afin de ne pas être affichée du tout, ou au contraire pour être dirigée vers un fichier texte, ou de log par exemple :

```
$ echo 'bonjour'
bonjour
$ echo 'bonjour' > /dev/null
$ echo 'bonjour' > text.txt
$ cat text.txt
Bonjour
```

Nota : rediriger une sortie vers un fichier inexistant aura pour effet de créer ce fichier.