SCR.1.1 TP 02 ⊥ :

Code de sortie des commandes

Formes de lancement des lignes de commandes

Redirections des entrées/sorties des commandes

Créer le répertoire `TP02` dans le répertoire `~/SCR.1.1`
Se placer dans `~/SCR.1.1/TP02/` et y créer le fichier `tp02-reponses.txt`
Créer les fichiers `fi` et `fifi` dans `~/SCR.1.1/TP02/`
Ainsi, `fi` et `fifi` sont présents dans le répertoire courant alors que `nofi` et `nofifi` n'y sont pas.

A lire.

1. Des codes de sortie (*exit status*) sont associés à la terminaison de chaque commande pour indiquer la manière dont la commandes s'est terminée. Linux associe un code de sortie égal à 0 pour une terminaison correcte ayant fourni un résultat correct. Les autres cas de terminaison génèrent un code de sortie de valeur différente de 0. La valeur exacte de cette valeur autre que 0 dépend de la ligne de commande et de la situation précise qui a conduit à cette terminaison. Par exemple, la situation `command not found` correspond à un code de sortie différent de `Permission denied`, différent de `No such file or directory`, etc. Les codes de sortie des commandes sont indiqués dans les pages du manuel des commandes. Par exemple, pour connaître les codes de sortie pour la commande `ls`, on va dans `man ls`

2. La variable `?` du shell contient le code de sortie de la ligne de commandes la plus récemment exécutée au premier-plan. Si `var` est une variable (du shell), la notation `$var` est remplacée par le shell par la valeur stockée dans `var`. Ainsi, `$?` est remplacée par le shell par le code de sortie de la ligne de commandes la plus récemment exécutée au premier-plan.
3. Pour chacune des commandes introduites dans l'énoncé, consulter les pages de son manuel si on ne sait pas ce que fait la commande. Les commandes (existantes) qui n'ont pas de page manuel directe sont probablement des commandes internes du shell. On y accède alors en faisant `man bash` et en allant au paragraphe `SHELL BUILTIN COMMANDS`.

I. Quelques formes des lignes de commandes.

Forme simple :

Pour chacune des lignes de commandes suivantes, interpréter les résultats et consulter les codes de sortie correspondants. Placer les réponses dans `tp02-reponses.txt`

1. Que fait la commande `echo` ? Consulter la page appropriée du manuel.
2. `echo "Hello world!"`
3. `echo "$?"`
4. `ls fi`
5. `echo "$?"`
6. `ls -l fi`
7. `echo "$?"`
8. `ls nofi`
9. `echo "$?"`

Pipeline :

Pour chacune des lignes de commandes suivantes, interpréter les résultats. En déduire ce que la notation `|` signifie pour le shell comme forme de lancement de commandes. Placer les réponses dans `tp02-reponses.txt`

1. `echo "Hello world1!" | echo "Hello world2!"`
2. `echo "Hello world2!" | echo "Hello world1!"`

Liste :

Pour chacune des lignes de commandes suivantes, interpréter les résultats. En déduire ce que signifient les notations `&&` et `||` pour le shell comme formes de lancement de commandes. Placer les réponses dans `tp02-reponses.txt`

1. Indiquer au préalable ce qu'on cherche à connaître en faisant suivre chacune des lignes de commandes par `; echo "$?"` et en déduire ce que signifie pour le shell le caractère `;` comme forme de lancement dans les lignes de commandes.
2. `ls fi ; echo "$?"`
3. `ls nofi ; echo "$?"`
4. `ls fi && echo "Hello world!" ; echo "$?"`
5. `ls nofi && echo "Hello world!" ; echo "$?"`
6. `ls fi || echo "Hello world!" ; echo "$?"`
7. `ls nofi || echo "Hello world!" ; echo "$?"`
8. `ls nofi || ls fi ; echo "$?"`
9. `ls fi || ls nofi ; echo "$?"`
10. `ls nofi && ls fi ; echo "$?"`

II. Redirections des entrées/sorties.

A lire.

Toute ligne de commandes lancée par le shell engendre un code qui se met en exécution dans la mémoire. On obtient alors ce qu'on appelle un processus. Lorsqu'un processus est engendré, le système lui alloue automatiquement une entrée standard, une sortie standard des résultats et une sortie standard des erreurs, associées toutes les trois, par défaut, au terminal de contrôle du processus. Ainsi, c'est vers son entrée standard qu'un processus "se tourne" s'il est en intéraction pour des données en lecture. Il se tournera vers sa sortie standard des résultats s'il a des résultats à écrire, et vers sa sortie standard des erreurs s'il doit écrire des erreurs. Ces entrée et sorties standards font partie de la table des descripteurs de fichiers du processus (les fichiers manipulés par le processus). Dans cette table, l'entrée standard occupe la ligne numéro 0. La sortie standard des résultats et celle des erreurs correspondent respectivement aux entrées (aux lignes) numéros 1 et 2 de cette table. Entrée et sorties standards peuvent être redirigées vers des fichiers dont le nom peut être choisi librement à la ligne de commandes.

Pour chacune des lignes de commandes suivantes, interpréter les résultats. En déduire, en particulier, ce que signifient pour le shell les notations `>, >>, 2>, 2>&1, <`, dans les lignes de commandes. Placer les réponses dans `tp02-reponses.txt`

1. `ls nofi fi`
2. `ls nofi fi > output_file`
3. `cat output_file`
   Que contient alors le fichier `output_file` ? En déduire ce qu'on a fait avec la ligne de commandes en 2. Placer la réponse dans `tp02-reponses.txt`
   Procéder de la même manière pour les situations suivantes, en plaçant dans `tp02-reponses.txt` ce qu'on en a déduit pour chacune des notations `>>, 2>, 2>>, 2>&1, <`
4. `ls nofi > output_file`
5. `cat output_file`
   Préciser alors dans `tp02-reponses.txt` ce que fait l'utilisation de `>` sur le contenu du fichier dont le nom est donné après `>`

6. `ls fifi nofifi > output_file`
7. `cat output_file`
8. `ls nofi >> output_file`
9. `cat output_file`
   Préciser alors dans `tp02-reponses.txt`, la différence avec ce qui s'est passé en 4 quant au résultat de la redirection `>>` sur le contenu du fichier dont le nom est donné après `>>`
10. `ls nofi fi 2> error_file`
11. `cat error_file`
12. `ls fi nofifi 2> error_file`
13. `cat error_file`
14. `ls nofi fi 2>> error_file`
15. `cat error_file`
    Préciser alors dans `tp02-reponses.txt`, ce que font les redirection `2>` et `2>>`
16. `ls nofi fi > output_file 2> error_file`
17. `cat output_file`
18. `cat error_file`
19. `ls nofi fi > output_and_error_file 2>&1`
20. `cat output_and_error_file`
    Préciser alors dans `tp02-reponses.txt`, l'intérêt d'utiliser `2>&1` comme mécanisme de redirection en soulignant la différence avec ce qu'on a fait en 16.
    Faire le lien avec ce qu'on a lu dans le paragraphe introductif à propos des numéros dans la table de descripteurs d'un processus.
21. Que fait la commande `wc` ? Consulter la page appropriée du manuel.
22. On tape `wc`, puis on appuie sur la touche `<CR>` (carriage-return : la touche retour). On tape ce qu'on veut : 2 ou 3 lignes, on revient à la ligne, on fait `CTRL-D`. Qu'obtient-on ?
23. `wc output_and_error_file`
24. `wc < output_and_error_file`
    Indiquer la différence entre les deux formes précédentes.
25. `ls -l /etc/ > output_file`
    Qu'a-t-on fait ? Placer la réponse dans tp02-reponses.txt
26. `wc < output_file`
    Qu'a-t-on fait ? Placer la réponse dans tp02-reponses.txt
27. En utilisant une des formes de lancement vues précédemment, donner une seule ligne de commandes sans le caractère `;` et réalisant le même objectif atteint par l'exécution de la ligne de commandes 25 suivie de celle de 26. Placer la réponse dans tp02-reponses.txt