POUGNES PHOENIX TSP

CSC 3102 - Introduction aux systèmes d'exploitation

Cette fiche n'est évidemment pas exhaustive.
Comme en Shell, les commentaires de cette fiche commenceront avec le caractère
(et seront colorés comme sur un éditeur de texte décent).

Petite liste de choses qu'on doit savoir écrire

#!/bin/bash : Première ligne de tout script shell. Signale à l'ordinateur l'interpréteur de commande que l'on s'apprête à utiliser. Bash est le shell par défaut, trouvable sur n'importe quelle machine sous UNIX.

echo : Renvoie l'argument vers la sortie standard (l'écran par défaut, sauf en cas de redirection)

caractères spéciaux : Doivent être précédés d'un \.
Les caractères spéciaux sont : \ ' " > < $ # * ? ; ( ) espace { } ‘ [1]

cat : Affiche le contenu du fichier en argument (d'un coup).

more : Affiche le contenu du fichier en argument page par page.

chmod : chmod $\alpha * \beta$, avec α=u,g,o, *=r,w,x, β=r,w,x.
Pour attribuer des droits en lecture/écriture/exécution (r/w/x) à l'utilisateur/le groupe/les autres (u/g/o)
Exemple : la commande "chmod u+x plop.sh" ajoute à l'utilisateur le droit d'exécution sur le fichier plop.sh .

a = x : Déclare une variable a en lui attribuant x comme valeur initiale.

$a : Désigne le contenu de la variable a.

read a : Saisie à partir de l'entrée standard (le clavier par défaut, sauf en cas de redirection) de la valeur qui sera stockée dans la variable a.

1. accent grave

Conditions et boucles

Conditions

Conditions

```
    # Condition classique
if <condition1> ; then
    <cmd1>
elif <condition2> ; then # not equal
    <cmd2>
    ...# les "elif" sont à répéter autant de fois que nécessaire.
else
    <cmd3>
fi
    # Case - Équivalent à une condition avec multiples "elif". Cherche var dans les motifs.
case var in
    motif1) corps1 ; ;
    motif2) corps2 ; ;
    ...
    motifn) corpsn ; ;
esac
```

Boucle

```
    # Boucle "While"
if <condition1> ; do
    <corps>
done

# Boucle "For" - On reste dans la boucle tant que var prend une des valeurs de la liste
for var in list ; do
    <corps>
done
```

Tests

Ces tests renvoient le booléen "vrai" si la condition testée est réalisée. Attention aux espaces qui rendent cette syntaxe affreuse...

Sur des valeurs

Soit a,b les varibales à comparer.
[a -eq b] : a = b
[a -ne b] : a ≠ b # ne, not equal
[a -gt b] : a > b # gt, greater than
[a -ge b] : a ≥ b # ge, greater or equal
[a -lt b] : a < b # lt, lower than
[a -le b] a ≤ b # le, lower or equal

Sur des chaînes de caractères

Soit chaine, chaine1, chaine2 les chaînes de caractères à tester/comparer.
[chaine1 = chaine2] : Vérifie si les chaînes sont identiques
[chaine1 != chaine2] : Vérifie si les chaînes sont différentes
[-z chaine] : Vérifie que chaine est vide
[-n chaine] : Vérifie que chaine est non vide

Arguments sur la ligne de commande

$i : Désigne le $i^{\text{ème}}$ argument de la commande, sachant que $0 désigne la commande elle-même.

$# : Désigne le nombre de paramètres.

$@ : Désigne la liste de l'ensemble (ordonné) des paramètres.

$(cmd) : Force l'interprétation de la commande cmd.
C'est différent de $cmd, qui désigne la valeur contenue dans la variable cmd

shift : Décale la liste des paramètre : $\forall i, \$i = \$(i+1)$ # le = désigne l'affectation ici

Comment survivre dans l'arborescence - commandes de base

pwd : Print Work Directory : Affiche le chemin absolu du repertoire de travail actuel.

cd : Change Directory : Pour se déplacer vers le répertoire passé en argument (fonctionne comme ls).

ls <chemin> : List Sort : liste le contenu de l'entrée en argument (répertoire courant sans argument).

. désigne le répertoire courant.

.. désigne le repertoire parent.

ln *data link* : Crée un lien dont le chemin est "link" vers l'entrée "data". le lien a le même inode, si l'un est supprimé, l'autre perdure

ln -s *data link* : Crée un lien symbolique dont le chemin est "link" vers l'entrée "data". Même contenu mais inodes différents. Si data est altéré, link est brisé.

touch *it* : Crée un fichier vide "it". [2]

mkdir : Make Directory. Crée un répertoire. *rmdir* sert à les supprimer s'ils sont vides.

rm : Remove. Pour les fichiers, voire les dossiers avec l'option -r

cp *source copie* : Copie.

mv *avant après* : Déplace. Ce procédé sert aussi à renommer les entrées.

Affichage de fichier

more : Affichage ligne par ligne du fichier en argument.

cat : Affichage complet du fichier en argument.

head -n : Affiche les n premières lignes du fichier.

tail -n : Affiche les n dernières lignes du fichier.

wc : Compte le nombre de lignes, mots et caractères d'un fichier.

sort : Trie les lignes par ordre lexicographique. En options, -n pour un tri numérique, -k2,4 pour un tri selon les champs 2 à 4 par exemple, -t 'α' pour changer le séparateur par le caractère α.

grep *motif fichier* : affiche les ligne de "fichier" contenant le motif "motif"

Redirections

a > b : Redirige a dans b. Si a est le résultat d'une commande, son contenu écrase celui de b.

a > > b : Redirige a dans b sans en écraser le contenu : si a est le résultat d'une commande, son contenu est placé à la suite de celui de b. # C'est différent de $cmd, qui désigne la valeur contenue dans la variable cmd

a < b : Met le contenu de b en argument de la commande a.

2. Si le fichier existe déjà, met à jour la date de dernière modification.

Processus

ps : Affiche les processus en cours.

pstree : Affiche l'arborescence des processus.

ctrl + z : Suspend le processus en cours. **fg** et **bg** permettent respectivement de le reprendre en *foreground* (avant-plan) ou **background** (arrière-plan)

ctrl + c : Stoppe un processus en *foreground*.

kill *n° PID* : Arrête un processus.# PID = Processus IDentifier

nice -n *priorité commande* : Lance le programme avec la priorité en option

nice -n *priorité n° PID* : Change la priorité d'un processus.

wait : Attend la fin du processus.

sleep x : Attend x secondes

Tubes

mkfifo *pipe* : Crée un tube nommé "pipe".# Il existe aussi des tubes anonymes, les "|".

echo *truc* > *pipe* : Écris "truc" dans le tube "pipe".

read *a* < *pipe* : Place le contenu du tube "pipe" dans la variable "a".

Verrous

Remarque. *Il ne faut pas oublier d'importer les scripts P.sh et V.sh avant de modifier les droits d'exécution.*

P.sh <nom> : Une fois verrouillé, le processus attend de retrouver un verrou <nom>.

V.sh <nom> : Déverrouille.

Remarque. *On place toujours les verrous dans le même ordre.*