DM 2 - ASR7 Programmation Concurrente

Rappels Système

Septembre

I Des commandes et une logique à connaître

— À quoi sert `which` ?

Il y a la page man...

— Quels sont les ensembles d'utilisateurs pouvant exécuter la commande `which` ?

`which which` qui nous donne à priori `/usr/bin/which` puis `ls -l /usr/bin/which` qui retourne quelque chose de similaire à `-rwxr-xr-x 1 root root 23504 6 déc. 2017 /usr/bin/which`. Ainsi, `root`, le groupe `root` et tous les utilisateurs peuvent exécuter la commande. Celui connaissant ses outils aura directement fait `ls -l $(which which)`

— Quelle est la différence avec `whereis` ?

Il y a la page man. Il suffit de comparer le descriptif...

— Peut-on utiliser la commande `locate` à la place de `which` ?

Elle retourne à priori plus d'informations que ce nous voulons avec `which` (pas seulement des commandes). Il faudra donc filtrer/trier toutes ces informations (essayez `locate which`...) ! En plus elle prendra le temps de s'exécuter en conséquence. Par ailleurs, ça nécessite que le ou les packages contenant `locate` et `updatedb` soient installés, et la base de données à jour. Donc non, on ne l'exécutera pas "à la place"...

— Que fait la commande `dmesg` ?

Celui qui a exécuté la commande directement pour voir le résultat est un fou furieux : c'est l'équivalent de "exécuter `rm *` juste pour voir", bref une "roulette russe". Celui qui n'est pas convaincu peut s'essayer avec la commande `hdparm` et pourra ainsi griller son disque dur pour s'en convaincre...

— Au vu des informations retournées par `dmesg`, que certains pourraient penser sensibles (donc à sécuriser), est-ce qu'un simple utilisateur peut utiliser la commande ?

Certains ont peut être répondu ce qu'ils pensaient, sans vérifier. C'est mal ;) Celui qui a exécuté la commande directement ne comprend pas la logique qu'on essaye généralement de mettre dans les exercices, notamment d'examen : il convient ici d'utiliser `which`, *etc.*

— La commande `top` m'indique que `firefox` utilise du CPU alors qu'il est démarré mais que je ne l'utilise pas. Comment faire pour stopper firefox ?

Rappel d'ASR5 : utilisation de la commande `kill` qui **permet d'envoyer un signal** à un processus dont on connait le `PID` (processus ID).
`top` indique le `PID`, mais on peut l'obtenir aussi avec `ps aux | grep firefox`. Il suffit de faire ensuite `kill -s SIGSTOP PID`.

— Je souhaite avoir deux commandes : `Fst` et `Fc`, la première permet de stopper `firefox`, l'autre permet de lui faire redémarrer son exécution. Comment faire ?

Du `ps aux | grep firefox`, du `cut` pour avoir le `PID`, des `kill` bien écrits avec les bonnes options, dans l'alias correspondant.

— On souhaite obtenir les informations du noyau concernant les cartes réseaux. Comment faire ?

Réseau en anglais, c'est network.
Donc `dmesg | grep net` devrait fonctionner...
Une autre solution est d'utiliser la commande `lspci` qui liste un peu plus de périphériques.

— On souhaite obtenir les informations du noyau concernant les disques durs, voire les espaces de stockage. Comment faire ?

En anglais, disque s'écrit "drive" et dur, "hard".
On pourrait essayer avec stockage, qui s'écrit "storage", mais... :)
Avec un peu de culture générale Linux, on sait que les disques sont des *devices* apparaissant dans `/dev/` par un nom commençant par sd (sda, sdb, *etc.*).
On peut aussi faire un grep sur la métrique de stockage, le GB.

— Je souhaite faire un *backup* de mon répertoire `$HOME/Work`. Comment faire ?

En utilisant correctement la commande `tar` avec les options `cvf`

— Faire un script `Mybckup` qui permet de réaliser le *backup* précédent. Attention, il faut préserver toutes les sauvegardes !

Si le fichier existe, il faut d'abord renommer le fichier existant en `bckp1.tar` par exemple. Mais ce fichier existe peut-être lui-aussi ! L'algo : si fichier `bckp1.tar` existe, teste

si `bckp2.tar` existe, *etc.* (boucle tant que), et à la fin, on renomme `bckupn.tar` en `bckupn+1.tar` et on redescend jusqu'à `backup1.tar`.
Puis on sauve.

— Que fait la commande `date` ?

Il y a la page man...

— Quelles options utiliser avec `date` pour obtenir le format `YYYY_MM_DD` ?
— Comment faire son `tar` en utilisant `date` dans le nom du fichier obtenu ?
— Comment mettre un job `cron` en place pour que toutes les semaines, le dimanche soir à 22h, un *backup* soit réalisé ?

On trouve facilement sur le net les informations pour remplir la `crontab`...

— La machine qu'on vient de configurer pour exécuter une tâche de façon périodique peut être arrêtée à 21h : son propriétaire peut préfèrer lire quelque chose comme Terry Pratchett ou simplement se coucher plus tôt le dimanche soir. Il faudrait pourtant que le *backup* plannifié puisse avoir lieu au plus proche de l'hebdomadaire. Grâce à `anacron` [1], c'est possible : comment ?

Parmi les choses à retenir ici, le fait qu'`anacron` ne soit pas un démon, doive être lancé par exemple par `cron` – et réponde à notre besoin ;)

1. `https://fr.wikipedia.org/wiki/Anacron`