TRAVAUX PRATIQUES LINUX

17 février 2022

ENSEIGNANT : GHISLAIN PANDRY

Exercice 1

Écrire un shell script qui écrit sur sa sortie standard les messages suivants :

1. mon nom est xxx
2. je suis appelé avec yyy arguments qui sont : 111 222 333 444

(xxx sera remplacé par le nom sous lequel ce shell script aura été invoqué, yyy par le nombre d'arguments et 111, 222, etc. par les arguments en question). Quand ce script fonctionnera correctement, invoquez le avec les cinq arguments :

1. Bienvenue dans le monde Linux
2. puis avec un seul argument contenant la chaîne de caractères : Bienvenue dans le monde Linux

Exercice 2

En utilisant exclusivement les commandes *cd* et *echo*, écrire le shell script "*recurls*" réalisant la même fonction que la commande *ls R*. C'est à dire que la commande *recurls rep1* devra lister les noms de tous les fichiers et répertoires situés sous le répertoire rep1, y compris les sous-répertoires et les fichiers qu'ils contiennent. Une solution très simple consiste à rendre le script recurls récursif. (un shell script est récursif s'il s'invoque lui-même).

Exercice 3

L'espace disque est précieux ! Une idée pour économiser : Ecrire un script qui recherche dans toute mon arborescence tous les fichiers qui n'ont pas été accédés depuis un temps T et dont la taille est supérieure a MIN, et les comprimer par l'utilitaire gzip. T et MIN sont des constantes définies au début du script par des valeurs judicieusement choisies. Au fait, a quoi sert MIN ? Un tel script pourrait être lancé une fois par semaine.

Exercice 4

Écrire un script dont le nom est process permettant de copier dans un tableau la liste des processus de l'utilisateur exécutant ce script, puis afficher le nom de chaque processus.

Exercice 5

Écrire un shell script démontrant que le shell fils hérite de son père, mais que le père n'hérite pas de son fils.

Exercice 6

Écrivez un shell script qui cherche dans votre arborescence personnelle tous les fichiers de noms *core, *.tmp, a.out* qui n'ont pas été accédés depuis plus de 3 jours, les supprime et vous envoie la liste par mail. Ce script sera exécuté tous les jours à trois heures du matin, sauf les samedi et dimanche.

Exercice 7

Écrivez le shell script *killprog* qui permet d'envoyer le signal SIGKILL à un processus désigné non pas par son PID mais par son nom : *killprog bash*. Il faudra prendre garde au fait que plusieurs processus différents peuvent porter le même nom, en présenter clairement la liste, et toujours demander confirmation avant d'envoyer le signal.

Exercice 8

Vérifiez que les permissions de votre répertoire d'accueil et votre umask sont conformes aux relations de confiance que vous entretenez avec les autres utilisateurs. Faites cette vérification pour vos répertoires principaux et vos fichiers de démarrage.

Exercice 9

1. Lancer la commande sleep 9999 en arrière-plan.
2. Quels sont le PID et le numéro de travail (job) du processus sleep précédent ?
3. Suspendre le processus sleep précédent et vérifier son état.
4. Relancer maintenant le processus sleep et vérifier son état.
5. Tuer le processus sleep.

Exercice 10

1. Créer une copie de bash dans /usr/bin/hack avec le SUID positionné.

   Vérifier les droits sur /usr/bin/hack.
2. Rechercher tous les fichiers disposant des bits SUID et/ou SGID
3. Chercher le package fournissant le fichier /usr/bin/hack. Que peut-on en déduire ?
4. Créer le fichier vide /usr/bin/rootedoor.
5. Vérifier à l'aide de chkrootkit que le système ne possède pas de rootkits connus.

Exercice 11

1. Écrire un script shell permettant de compter le nombre de processus.
2. Ecrire un script shell qui affiche la liste des processus triées par ordre alphabétique.

Exercice 12

1. Quel est le rôle du shell ?
2. Comment crée-t-on, dans le répertoire courant, le répertoire ici en ligne de commande sous Linux ?
    1. pwd ici

    2. cr ici
    3. rmdir ici
    4. mkdir ici
3. Parmi ces chemins, lequel est absolu ?
    1. ./usr/local/sbin
    2. Documents/perso
    3. /usr/local/sbin
4. À quoi sert le pipe | ?
5. Quels seront les résultats des commandes ls | grep -v documents et ls | grep -iv documents dans le répertoire contenant : Bureau, Documents, Images, Musique
6. Quelle commande permet de modifier la priorité d'un processus en cours d'exécution ?
    1. kill
    2. nice
    3. renice
    4. ps
7. Quelle doit être la première ligne d'un script shell ?

Exercice 13

1. Écrire un shell-script qui calcule la somme en kilo-octets de la mémoire utilisée par tous les processus de l'utilisateur. Le champ size va de la 26ème à la 29ème position. L'option -cm-n de la commande cut permet de récupérer les caractères allant la $m$ème à la $n$ème position incluse.
2. Écrire un shell-script qui affiche l'utilisateur consommant le plus de cpu ainsi que son pourcentage total d'utilisation.
3. On désire écrire une commande de manipulation d'une poubelle de fichiers, can.
   Pour ce faire, on dispose d'un répertoire $\$HOME/.poubelle$ qui contiendra l'ensemble des fichiers mis à la poubelle. Si ce répertoire n'existe pas, la commande can doit le créer. La commande can accepte les trois options suivantes :
   — can -l liste la poubelle ;
   — can -r purge la poubelle ;
   — can filenames transfert les fichiers filenames dans la poubelle.

Exercice 13

1. Ecrire une commande qui renomme les entrées d'un répertoire. Pour chacune des entrées du répertoire, l'utilisateur doit saisir un nouveau nom s'il souhaite renommer l'entrée ; l'entrée est alors renommée. Cette commande doit :

— afficher chaque entrée du répertoire courant (si la commande est appelée sans argument) ou d'un répertoire passé en argument (en vérifiant que celui-ci existe) ;

— lire la réponse de l'utilisateur : RETURN signifiera «pas de renommage du fichier correspondant» et toute autre réponse sera utilisée comme nouveau nom du fichier ;

— tester l'existence d'une entrée ayant le même nom que la réponse.

La commande mv f1 f2 écrase le fichier f2 si celui-ci existe. Dans ce cas, soit affichez un message et passez au fichier suivant, soit demandez un nouveau nom, soit demandez confirmation.

2. Écrire une procédure envoi nom fich qui envoie le contenu du fichier fich à l'utilisateur nom soit par write s'il est connecté et accepte les messages soit par mail. La commande devra tester la syntaxe d'appel (2 arguments), l'existence du fichier fich, la déclaration de l'utilisateur nom dans le fichier /etc/passwd (une entrée commençant par nom :).
3. Écrire une procédure svgdir r1 r2 qui effectue la copie du répertoire r1 (qui doit exister) sur un répertoire r2 (qui ne doit pas exister) en reconstruisant la même arborescence. Cet exercice fait intervenir la récursivité. Deux méthodes sont possibles, soit se déplacer dans l'arborescence à copier, soit ne pas se déplacer.
4. On se propose d'écrire un script shell pour gérer un annuaire. Cet annuaire comporte une série de lignes, chacune composée de 5 champs : nom, prénom, numéro de téléphone, bureau et service. Les champs sont séparés par le caractère deux-points. Ce script shell est appelé avec une seule option pour réaliser chacune des fonctions ci-après :

   — afficher l'annuaire trié par service et par nom (-t) ;

   — afficher l'annuaire trié sur le numéro de téléphone (-T)

   — afficher la liste des noms des inscrits (-M)

   — afficher le dernier inscrit (-d)

   — afficher la liste des inscrits sous la forme Nom.Prénom (-m)

   — rechercher un inscrit à partir de son nom (-g bac) ou d'une partie seulement (-g bou) et sans distinction des majuscules/minuscules

   — ajouter un nouvel inscrit (-a)

   — créer un fichier par service (-e)

   — supprimer un inscrit (-s Bac)

   — lister le personnel d'un bâtiment (-b X). Le bâtiment est indiqué par la première lettre du bureau.

   — etc (selon votre imagination ou vos besoins).

   En outre, le programme doit respecter les consignes ci-après :

   — tester la syntaxe d'appel : il faut au moins un argument : le nom de l'annuaire et au maximum une option

   — tester si l'annuaire existe et est accessible

   — sans option, le script shell affiche simplement l'annuaire trié par nom

- pour l'ajout d'un inscrit, boucler en lecture clavier pour saisir successivement chacun des champs et demander confirmation avant de l'enregistrer
- pour la suppression, l'argument doit représenter exactement le premier champ
- être protégé contre les signaux TERM (N° 15) et INTR (N° 2 et touche CTRL C).

5. Écrire un script pour créer un compte utilisateur voir : man useradd Utilisez votre script de vérification d'existance d'utilisateur avant de creer. Il faudra vérifier que l'utilisateur en cours d'exécution est bien root voir echo $\$USER$ Il faudra créer son home dans /home après avoir vérifier qu'il n'y a pas déjà un répertoire portant le même nom. Il faudra répondre à une suite de question : voir man read
   - login
   - Nom
   - Prenom
   - UID
   - GID
   - Commentaires