université PARIS-SACLAY | FACULTÉ DES SCIENCES D'ORSAY

vendredi 20 septembre 2024

Introduction à l'informatique
L1

TP n° 2

Consignes les exercices ou questions marqués d'un ⋆ devront être rédigés sur papier (afin de se préparer aux épreuves écrites du partiel et de l'examen). En particulier, il est recommandé d'être dans les mêmes conditions qu'en examen : pas de document ni de calculatrice. Tous les TPs se font sous Linux.

1 Redirections et manipulation de fichiers textes

Exercice 1

⋆ Pour chacune des questions suivantes, donner les commandes permettant de réaliser les actions demandées. On cherchera d'abord les commandes en s'aidant du cours ou des pages de manuel.

1. Si vous êtes sur la session de secours, vérifier si un répertoire `IntroInfo` existe dans le rerépertoire personnel. Si c'est le cas, le renommer en `IntroInfo_old`.
2. Créer un répertoire `IntroInfo`.
3. Se placer dans le répertoire `IntroInfo`. Créer un sous-répertoire TP2 et se placer dedans.
4. Lister de façon détaillée le contenu du répertoire. Dire maintenant comment créer un fichier `liste.txt` contenant cette liste détaillée.
5. Lister de façon détaillée les fichiers `liste.txt` ainsi qu'un fichier `pasla.txt` (inexistant). Quel est l'affichage de la commande ?
6. Effectuer la même commande en redirigeant les erreurs vers un fichier `erreur.txt` et la sortie standard vers un fichier `liste2.txt`. Afficher tour à tour (en 2 commandes) le contenu de ces fichiers.

Exercice 2

Afficher au moyen de la commande `cat` le fichier `/etc/passwd` (ce dernier contient la liste des utilisateurs « locaux » de la machine, *i.e.* pas les utilisateurs dont les comptes sont en réseaux comme les profs ou les étudiants). Pour chacune des actions suivantes, on se référera au cours 1 et 2 et éventuellement à la page de manuel de la commande en question si le cours ne donne pas de détail [1]. On rappelle pour quitter une page de manuel et revenir au terminal, il suffit d'appuyer sur `q` (pour `quit`). Les flèches haut et bas permettent de se déplacer dans la page de manuel.

Remarque il est vivement conseillé d'ouvrir un fichier texte et d'y copier/coller les commandes que vous essayez afin de garder une trace de ce que vous avez fait.

1. afficher les 5 premières lignes du fichier `/etc/passwd`
2. afficher la page de manuel de la commande `tac`
3. utiliser la commande `tac` pour afficher le fichier `/etc/passwd` à l'envers
4. trier le fichier `/etc/passwd`. Quel est l'ordre utilisé ?
5. lire attentivement la page de manuel pour trier selon le troisième champ. On cherchera quelle option permet de spécifier le séparateur de champs et quelle option permet de donner un numéro de champ. Quel est l'ordre utilisé ?
6. trier selon le même champ que la question précédente mais en utilisant l'ordre numérique (chercher dans la page de manuel). En quoi l'ordre est-il modifié ? donner deux lignes dont l'ordre de l'une par rapport à l'autre a changé.

1. si la page de manuel n'est pas disponible sur votre machine, vous pouvez consulter l'url suivante : `http://www.linux-france.org/article/man-fr/man1/Index-1.html`

Exercice 3

Récupérer le fichier se trouvant à l'adresse suivante :

`https://www.lri.fr/~kn/teaching/ii/td02/cdm.txt`

(en utilisant un navigateur Web) et le sauver dans le repértoire `TP2`. Afficher le contenu du fichier. Ce dernier contient des listes de matches de phase finale de coupe du monde, sous la forme :

`PAYS1:PAYS2:GROUPE`

(où `GROUPE` est une lettre entre `A` et `H`). On souhaite déterminer combien d'équipe distinctes sont dans le fichier.

1. lire la page de manuel de la commande `cut`. Déterminer comment extraire uniquement la première « colonne » du fichier `cdm.txt` (*i.e.* déterminer comment afficher uniquement le premier champ de chaque ligne, en spécifiant que le délimiteur de champ est «`: `»)
2. Sauver le résultat de la première commande dans un fichier `liste1.txt` on peut sauver la sortie d'une ligne de commande en écrivant `commande arg1 ... argn > fichier`. Sauver de la même manière la deuxième colonne du fichier `cdm.txt` dans un fichier `liste2.txt`
3. sachant que le « `|` » permet d'enchaîner deux commandes en passant la sortie de la première comme entrée à la seconde, utilisez dans un premier temps la commande `cat` pour afficher les fichiers `liste1.txt` et `liste2.txt` l'un à la suite de l'autre puis enchainer le résultat avec la commande `sort`.
4. consulter (encore) la page de manuel de `sort` pour trouver l'option qui permet de supprimer les doublons. Afficher la liste des pays sans doublons
5. consulter la page de manuel de la commande `wc` (*word count*) pour savoir comment compter le nombre de lignes d'un fichier. En déduire comment rajouter la bonne invocation de la commande `wc` à votre enchainement pour déterminer le nombre de pays.

Exercice 4

La commande `dc` implémente une calculatrice en notation *polonaise inversée* (les opérandes sont placées sur une pile et les opérations dépilent les opérandes et empilent le résultat). Elle attend des ordres de calcul sur son entrée standard et les exécute. Par exemple (les lignes en *italique* sont celles à taper au clavier, les lignes en **gras** sont celles affichées par le programme) :

```
$ dc
42                ← invocation de la commande
41                ← place l'entier 42 sur la pile
+                 ← place l'entier 41 sur la pile
p                 ← dépile deux nombres et place leur somme sur la pile
 83               ← affiche le sommet de pile (print)
```

1. au moyen d'un éditeur de texte, créer un fichier `calculs.txt` contenant les lignes :

```
40
260
+
10
/
p
12
*
p
```

2. ⋆ Que serait l'affichage produit si cette série de commandes étaient données à `dc`. Donner *deux* lignes de commandes différentes permettant de passer ce fichier en entrée à `dc`.

2 Gestion des processus

Sous Unix, les processus peuvent avoir l'état suivant :

```
R       en cours d'exécution
D       en attente, non interruptible (généralement en train de faire une entrée/sortie)
S       en attente, interruptible
T       arrêté
Z       « zombie », processus défectueux, terminé mais dont les ressources ne sont pas encore libér
```

Il peut de plus y avoir des informations complémentaires :

```
+       le processus est en avant plan
s       le processus est un leader de session (usuellement c'est le shell)
l       le processus est en fait un thread (sous-processus)
```

1. Tapez la commande `ps u`. Quels sont les programmes en cours d'exécution et quels sont leurs états.
2. Exécuter la commande `glxgears`, sans la placer en tâche de fond. Constater que cette commande fait deux choses :
   — Elle ouvre une fenêtre graphique dans laquelle une animation 3D basique s'exécute
   — Elle affiche toutes les 5 secondes des statistiques sur sa sortie standard, notamment le nombre d'images par secondes de l'animation (FPS signifiant *frame per second*).

   Cette commande permet de tester le bon fonctionnement de la carte graphique de la machine.

   ⋆ Expliquer par quelles commandes vous pouvez interrompre le processus `glxgears` pendant un temps arbitraire, puis lui faire reprendre son exécution. Attention, on ne souhaite pas utiliser les raccourcis clavier et commandes tels que `CTRL-z` ou `fg`/`bg`, mais plutôt procéder par envoi de signaux.
3. Utiliser un second terminal pour interrompre la commande `glxgears` pendant au moins 10 secondes. Quel est l'impact sur la vitesse d'affichage du programme ?
4. Le programme `glxgears` fonctionne de la façon suivante :
   — Sauvegarde de l'heure système dans une variable `t0`
   — Dessin continu de l'animation. À chaque fois que la fenêtre est entièrement redessinée, le programme incrémente un `compteur`.
   — Sauvegarde de l'heure système dans une varible `t1`. Si `t1 - t0 >= 5s`, alors calcule et affiche `compteur/(t1 - t0)`.

   Expliquer pourquoi le nombre de FPS diminue lorsque le programme est en pause.

3 Scripts *bash* (bonus)

3.1 Exercice 4

On rappelle que la syntaxe d'un test en shell est :

```
if commande
then
    ...
else
    ...
fi
```

ou

```
if commande
then
    ...
fi
```

si on souhaite ne pas mettre de bloc `else`.

De plus, il est possible d'enchaîner plusieurs blocs `if/else` sans les imbriquer en utilisant l'instruction `elif`. Ainsi

```
if commande1
then
    ...
elif commande2
then
    ...
elif commande3
then
    ...
else
    ...
fi
```

est équivalent à

```
if commande1
then
    ...
else
    if commande2

    then
    ...
    else
      if commande3
      then
         ...
      else
        ...
        fi
    fi
fi
```

Écrire un script shell `stats.sh` prenant en argument un chemin p et effectuant les affichages suivants (on lira attentivement la page de manuel de la commande `test`) :

— s'il n'y a pas d'argument passé sur la ligne de commandes (c'est à dire si la chaine `"$1"` est vide) afficher un message d'erreur indiquant qu'il manque un argument et terminer

— si le chemin p ne dénote pas un fichier ou répertoire existant, afficher un message d'erreur et terminer

— si le chemin p ne dénote pas un fichier ou répertoire lisible (c'est à dire pour lequel vous n'avez pas les droits en lecture), afficher un message d'erreur et terminer

— si le chemin p dénote un fichier, afficher le message :

```
Nombre de lignes dans le fichier :
xx
```

où `xx` est le nombre de lignes du fichier

— si le chemin p dénote un répertoire, afficher le message :

```
Nombre d'entrées dans le répertoire :
xx
```

où `xx` est le nombre d'entrées (fichiers ou répertoires) contenu dans p. On demande juste le nombre d'entrées contenu au niveau inférieur, pas les entrées se trouvant dans des sous-répertoires de p.

— Dans tous les autres cas afficher :

```
Type de fichier non supporté
```