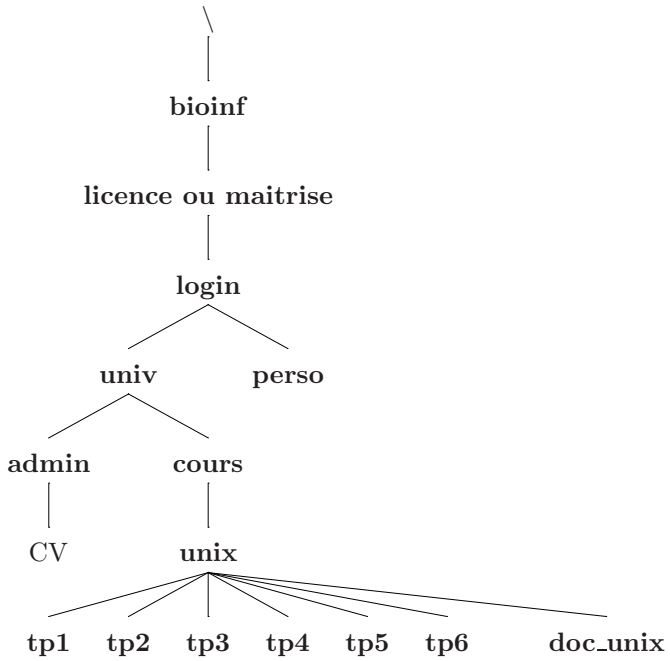
Mise à niveau UNIX
Licence Bio-informatique

TP6 - Révisions

Unix

Exercice 1 (Arborescence)**.**



1. Copiez, déplacez, renommez, nettoyez votre arborescence pour qu'elle est la structure ci-dessus (les répertoires apparaissent en gras).
2. Allez chercher les documents du cours sur la page du cours dont l'adresse est précisée ci-dessous. Sauvegardez dans le répertoire `doc_unix`.

   `http ://www.pps.jussieu.fr/ jch/enseignement/bioinfo`

3. Créez une archive compressée contenant votre répertoire `unix` et envoyez-la à l'aide de la commande `pine` à votre chargée de TP dont l'adresse est `tasson@pps.jussieu.fr`

Exercice 2.

Pour vérifier le contenu de votre arborescence, vous pouvez utiliser la commande `ls -R ~ > arborescence.txt`. Expliquez la signification de cette commande. A partir de votre répertoire `TP6`, testez cette commande. Vérifiez que le fichier à bien été créé à l'aide de la commande `less`.

Exercice 3.

1. Connectez vous via SSH en utilisant le login `tassonc` sur le serveur `nivose.informatique.univ-paris-diderot.fr`.
   Vous utiliserez le mot de passe (provisoire) `abc$123`.
2. Une fois connecté, vous trouverez un dossier `correction_unix` qui contient une archive compressée `correction.tar.gz`. Sans vous déconnecter et en utilisant la commande `scp`, copiez dans votre répertoire `unix` le dossier `correction_unix` (on oubliera pas l'option `-r` de `scp`).
3. Déconnectez-vous.

Exercice 4.
Placez-vous dans le répertoire `correction_unix` et décompressez l'archive en utilisant la commande `gzip -d correction.tar.gz | tar xf -`. Expliquez la signification de cette commande.

Exercice 5.
Pour ceux qui travaillent à deux sur une machine, copiez l'ensemble de votre arborescence chez votre binôme en vous servant de la commande `scp`.
N'hésitez pas à demander l'assistance de votre chargée de TP.

Exercice 6.
En vous plaçant dans votre répertoire personnel, modifiez les droits de lecture du répertoire `perso`, vous seul devez avoir les droits d'écriture, de lecture et d'exécution sur ce répertoire.
Indication : Utilisez la commande `chmod [ugo+/-rwx]`.

Exercice 7.
En vous plaçant dans votre répertoire personnel, créez un lien symbolique appelé `tp6_lien` vers votre répertoire `tp6`.
`tp6_lien` est maintenant un synonyme de `/univ/cours/unix/tp6`.
Indication : Utilisez la commande `ln -s lien cible`.

Exercice 8.

1. Placez vous dans votre répertoire `tp6`. A l'aide de la commande `emacs test.txt`, créez un fichier `test.txt` dans lequel vous écrirez une phrase de votre choix (le symbole & à volontairement été omis).
2. Sans quitter `emacs`, revenez dans le shell. A l'aide de la commande `C-Z` endormez le processus `emacs`. Puis réveillez le en utilisant la commande `fg` qui le fait passer en tache courante.
3. Sans quitter `emacs`, revenez dans le shell. A l'aide de la commande `ps` déterminez le numéro (PID) du processus `emacs` que vous venez de lancer. A l'aide de la commande `kill PID` tuez ce processus.
4. Ouvrez le fichier `test.txt` à l'aide d'`emacs`. Sans quitter `emacs`, revenez dans le shell. Tuez le processus `emacs` en utilisant la commande `xkill` et en désignant la fenêtre du processus `emacs`.

Conditionnelles et boucles en C

Exercice 9.

1. `lect_carre.c` : écrire un programme affichant le carré d'un nombre lu au clavier.
2. `lect_double.c` : écrire un programme lisant deux nombres au clavier, et affichant leur somme et leur produit.
3. `somme_10.c` : écrire un programme calculant la somme 1 + 2 + ... + 10, et l'affichant à l'écran. On se servira d'une variable `somme` initialisée à 0, puis d'une boucle `for` allant de 1 à 10 : à chaque étape, la valeur courante du compteur sera ajoutée à `somme`.
4. `somme_n.c` : modifier le programme précédent pour calculer la somme 1 + 2 + ... + n, où n est un nombre quelconque lu au clavier. Afficher également la moyenne des nombres 1 à n, c'est-à-dire leur somme divisée par n. L'opérateur de division s'écrit `/`.
5. `moyenne.c` : un peu plus dur... écrire un programme lisant au clavier un entier n quelconque, puis lisant successivement n entiers, puis affichant la moyenne de ces entiers.

Exercice 10.

1. Ecrire un programme `sup_10.c` lisant un entier `n` au clavier, puis `n` entiers. Pour chacun des `n` entiers lus, le programme affichera, en fonction de sa valeur :
   – "`cet entier est plus grand que 10`", ou bien
   – "`cet entier n'est pas plus grand que 10`".
   Remarque. Plutôt que d'entrer ces nombres au clavier, vous pouvez les stocker au préalable dans un fichier `entiers.txt`, contenant un entier par ligne. Le premier entier indiquera la valeur de `n`, et sera suivi des `n` autres. L'exécution se fera en entrant la commande `cat entiers.txt | ./a.out`.
2. Le programme `detect_10.c` affichera, après la lecture des `n` nombres et en fonction des valeurs lues, "`au moins un de ces entiers est plus grand que 10`", ou bien "`aucun de ces entiers n'est plus grand que 10`".
   Indication : Déclarer en début de programme une variable `drapeau` initialisée à `0`. Après la lecture de `n` nombres, le drapeau doit être à `1` si l'un des nombres lus ètait plus grand que 10, et à `0` sinon.
3. Après la lecture des `n` nombres, le programme `max_min.c` affichera le plus grand des nombres lus. Plus dur : afficher aussi le plus petit.

Exercice 11.

Envoyez les fichiers `.c` que vous avez rédigés aujourd'hui à votre chargée de TP sous la forme d'une archive compressée.