

Travaux dirigés de Perl n°3

Perl langage 3

—École Ouverte Francophone—

Perl : références et application à l'administration système

Le but de ce TD est de continuer l'apprentissage du langage *Perl* par des exemples d'utilisation des références appliqués à l'administration système.

► Exercice 1. (Références premiers pas)

Construisez une référence vers une table de hachage dont les clefs seraient le login d'une personne et la valeur une référence vers une table de hachage comportant les clefs suivantes :

- `uid` : son numéro d'utilisateur,
- `home` : son répertoire personnel,
- `gid` : une référence vers un tableau comportant ses numéros de groupe.

Prenez par exemple :

- `root`, d'`uid` 0, son répertoire personnel est `/root` et ses groupes sont 0, 2 et 10,
- `paul`, d'`uid` 500, son répertoire personnel est `/home/paul` et ses groupes sont 200, 4 et 10.

Faites un schéma de cette structure en mémoire.

Affichez d'abord cette structure de données à l'aide du module `Data::Dumper`

Écrivez ensuite le code Perl permettant d'afficher toute la structure sans utiliser ce module (le format d'affichage est libre) : n'utilisez pas l'opérateur `ref`, mais utilisez le fait que vous connaissez les trois clefs des tables de hachages (`uid`, `home` et `gid`) ainsi que les types de leurs valeurs respectives.

Dans un premier temps, affichez la liste des numéros de groupe avec `foreach`, dans un second avec `join`. Affichez finalement chaque liste de numéros de groupe par ordre numérique.

► Exercice 2. (Références)

Le but de cet exercice est la manipulation du fichier `passwd`. Ce fichier est composé de 7 champs séparés par le caractère `:` (deux-points). L'ordre des champs est le suivant :

- `login` : le nom du compte de l'utilisateur,
- `passwd` : la séquence chiffrée du mot de passe (ou `*` dans certains cas),
- `uid` : le numéro de l'utilisateur (0 pour `root`),
- `gid` : le numéro de groupe de l'utilisateur (attention : il n'y a ici qu'un seul numéro, ne pas confondre avec ce qui vous a été demandé dans l'exercice précédent),
- `info` : des informations sur l'utilisateur (nom, etc),
- `home` : le répertoire personnel de l'utilisateur,
- `shell` : le chemin de l'exécutable de son shell préféré.

Voici les étapes de notre projet :

1. Quelle structure pourrait-on utiliser pour stocker en mémoire toutes ces données et les manipuler facilement ?
2. On se propose de stocker ces informations dans un tableau qui aura pour valeur des référence vers des tables de hachage dont les clefs seront `login`, `uid`, `gid`, `info`, `passwd`, `home` et `shell`. Faites un schéma de cette structure de données.
3. Écrivez une fonction `parse` qui prend en paramètre le nom du fichier à analyser. Elle renvoie une référence vers ce tableau et elle s'utilisera de la sorte : `my $ref = parse('/etc/passwd');`
4. Écrivez une fonction `dump` qui effectue l'affichage de toutes ces informations. Chaque utilisateur sera affiché sur une seule ligne au même format que les lignes du fichier `/etc/passwd` et les utilisateurs seront affichés dans l'ordre de leur présence dans le fichier (pas de tri à faire, on respecte l'ordre du tableau).
5. Dans le programme principal après l'appel à la fonction `parse`, modifiez vos données des manières suivantes :
 - 5.1. ne gardez que les comptes dont les `uid` sont des nombres pairs (`%` et `grep`),
 - 5.2. ajoutez 10 à tous les `gid` (`map`),
 - 5.3. triez le tableau par ordre alphabétique du `login` (`sort`).Utiliser votre fonction `dump` pour vérifier.