Deux douzaines de questions pour commencer avec PERL

Linux Gazette n°101 — Avril 2004

Arun Mahadevan

Table des matières

Ces questions ont été recueillies à partir de celles réellement posées lors d'une interview sur Perl. Elles sont organisées de manière à pouvoir aider quiconque débute en Perl.

1. Quelle est la première ligne d'un script Perl ?

La première ligne d'un script Perl donne le chemin vers un exécutable Perl.

Par exemple : `#!/usr/bin/perl`

Cette ligne indique à UNIX quel programme utiliser pour lancer effectivement le script. Cette ligne doit correspondre au chemin réel de l'interpréteur Perl.

2. Faut-il passer des arguments/paramètres lorsqu'on invoque Perl ? Quels arguments/paramètres utilisez-vous lorsque vous invoquez Perl et quelle est leur signification ?

Oui, il est possible de passer des arguments/paramètres lorsqu'on invoque Perl. Par exemple, `#!/usr/bin/perl -w` ordonnera à Perl d'activer les messages d'avertissement.

Les arguments/paramètres importants incluent :

- `-c` : vérifie seulement la syntaxe (lance les blocs BEGIN et END)
- `-d` : lance le script avec le débogueur
- `-P` : exécute le préprocesseur C sur le script avant la compilation
- `-w` : active les messages d'avertissement pour la compilation des scripts

perldoc perlrun ou **man perlrun** donnent une page de manuel contenant une liste complète des arguments.

3. Quelle est la valeur d'une variable non initialisée en Perl ?

La valeur d'une variable non initialisée en Perl est `null`.

4. Quels sont les différents types de variables en Perl ?

Il existe trois types de variables en Perl. Ce sont :

1. Les scalaires
2. Les tableaux
3. Les tableaux associatifs (hash)

5. Qu'est-ce qu'une variable scalaire ?

Une variable scalaire contient une variable unique. Une variable scalaire peut contenir un nombre, une chaîne de caractères ou une référence. On accède aux scalaires en préfixant son identifiant par `$`.

6. Quelles valeurs peut-elle contenir ?

Une variable scalaire peut contenir un nombre, une chaîne de caractères ou une référence. Par exemple :

7. Qu'est-ce qu'un tableau ?

Les tableaux de Perl sont des collections ordonnées de scalaires. Un nom de tableau commence par un arobase (`@`). Il n'est pas nécessaire que les éléments d'un tableau soient du même type.

Par exemple :

Comme en C, les crochets `[ ]` sont utilisés pour faire référence aux éléments, `$a[6]` est l'élément à l'index 6 dans le tableau `@a`. Comme en C, les index de tableaux commencent à 0.

8. Que sont les tableaux associatifs ?

Les tableaux associatifs sont une structure de données clé/valeur fournie par Perl. Les tableaux associatifs sont optimisés pour trouver la valeur d'une clé très rapidement. Les variables de tableaux associatifs commencent par un signe pourcent (`%`) et utilisent les accolades `{ }` pour les accès à une clé particulière. Par exemple :

Il est souvent plus lisible d'utiliser l'opérateur => entre les paires clé/valeur.

9. Quelle est la différence entre un tableau et un tableau associatif ?

Les tableaux sont utilisés pour enregistrer une liste de scalaires, tandis que les tableaux associatifs sont utilisés pour enregistrer des paires clé/valeur.

Les tableaux sont indexés par des entiers (de 0 à N), tandis que les tableaux associatifs sont indexés par des chaînes de caractères.

Les tableaux utilisent les crochets pour faire référence à un élément (par exemple, `$a[4]`) tandis que les tableaux associatifs utilisent les accolades pour récupérer la valeur en utilisant sa clé (par exemple `$h{"clé"}`).

10. Quel est le type des données des clés d’un tableau associatif ?

Les clés d’un tableau associatifs sont du type de données « string » (chaîne de caractères).

11. Comment récupérer toutes les clés d’un tableau associatif ?

Il est possible d’utiliser la fonction `keys` pour récupérer une liste des clés d’un tableau associatif. Par exemple :

12. Comment récupérer toutes les valeurs d’un tableau associatif ?

Il est possible d’utiliser la fonction `values` qui retourne une liste des valeurs d’un tableau associatif. Par exemple :

13. Comment trier les clés d’un tableau associatif ?

Il est possible d’utiliser la fonction `sort` pour trier la liste des clés d’un tableau associatif. Par exemple :

retourne une liste triée des clés du tableau associatif `%map`.

14. Comment fonctionne `sort` sur les valeurs 2, 10 et 1, et dans quel ordre sont-elles triées en temps normal ?

La fonction `sort` trie les éléments dans l'ordre lexicographique. Donc si nous avons une liste contenant les valeurs 2, 10 et 1, la sortie du tri sera 1, 10, 2 (parce que 10 vient avant 2 dans l'ordre lexicographique).

15. Comment passer outre ce problème et s'assurer que le tri se passe correctement et que le résultat final est 1, 2, 10 ?

Pour trier les éléments d'une liste dans l'ordre numérique nous devons l'expliciter spécifiquement dans la fonction de tri.

16. Comment obtenir le nombre d'éléments d'un tableau ?

Nous pouvons obtenir le nombre d'éléments en évaluant le tableau dans un contexte scalaire, par exemple :

Nous pouvons aussi spécifier explicitement ceci en utilisant la fonction `scalar` comme ci-dessous :

17. Quel est le moyen le plus rapide d'accéder à la valeur du dernier index d'un tableau ?

La valeur du dernier index d'un tableau peut être obtenue grâce à `$#`. Par exemple :

18. Quel est le moyen le plus rapide d'accéder au premier élément d'un tableau (sans utiliser les crochets `[]`) ?

On peut utiliser la fonction `shift` pour récupérer le premier élément d'un tableau. Par exemple :

19. Qu'est ce que `unshift` ?

`unshift` fait l'opération inverse de `shift`. Il ajoute une liste au début d'un tableau, et retourne le nouveau nombre d'éléments du tableau.

La syntaxe est : `unshift TABLEAU,LISTE;`

Par exemple :

20. Dans le bloc principal, quelle est la valeur par défaut à laquelle accède `shift` ?

Si `TABLEAU` est omis, la fonction `shift` s'applique au tableau `@_` (c'est-à-dire la liste d'arguments passée à une sous-routine). Dans la portée du fichier (dans le flot principal du programme), `shift` sans argument s'applique à `@ARGV` (c'est à dire la liste des arguments de la ligne de commande). Par exemple :

21. Comment appeler une sous-routine ? Comment passer des paramètres ?

Une sous routine peut être définie comme ci-dessous :

Cette sous-routine peut être appelée depuis le programme principal par :

ou

Le passage de paramètres est similaire à ce qui se fait en C, par exemple :

22. Comment accéder aux paramètres passés à une sous-routine ? C'est-à-dire où sont ils enregistrés ?

On peut accéder aux paramètres passés via le tableau @_. Par exemple :

23. Que se passe-t-il quand on passe un tableau associatif suivi de trois valeurs scalaires en paramètres d'une sous-routine ?

Le tableau associatif et les trois variables passeront dans la sous-routine dans une longue liste dans le tableau `@_`.

La valeur du tableau associatif ne peut pas être récupérée correctement dans la sous-routine puisqu'elle est convertie en une liste.

Lorsque vous passez plus d'une variable incluant un tableau ou un tableau associatif, il est conseillé de passer les variables par référence.

24. Comment accéder aux variables d'environnement UNIX dans un script Perl ?

Les variables d'environnement UNIX sont accessibles dans Perl via le tableau associatif `%ENV`. Par exemple :

25. Le mot de la fin

Merci de me faire suivre vos suggestions/retours (en anglais). Merci, Arun.

Paru dans le n°101 de la Linux Gazette d'avril 2004.

Traduction française par Isabelle Hurbain `<isabelle POINT hurbain CHEZ pasithee POINT net>`.

Relecture de la traduction française par Joëlle Cornavin `<jcornavi CHEZ club TIRET internet POINT fr>`.