Perl FAQ

Date de publication : 24/06/2005

Dernière mise à jour : 16/03/2012

Bienvenue sur la FAQ Perl. Cette FAQ a pour vocation de vous enseigner ou de vous faire revoir les notions élémentaires de ce fantastique langage. Perl est très utilisé dans différents domaines depuis la gestion système, le réseaux, l'administration de bases de données, le web (CGI), la bureautique, la conception d'interfaces graphiques ou des contextes scientifiques telle la bioinformatique. Nous espèront que cette FAQ vous sera d'une grande utilité.
Vous souhaitez participer à l'amélioration de cette FAQ, n'hésitez pas !!
Bonne lecture !

Ont contribué à cette FAQ :

Djibril (Site personnel) - Woufeil (home) - Stoyak - Philou67430 - Jedai - GLDavid - 50Nio - 2Eurocents - Schmorgluck - Alek-C - Jasmine80 - kuzco - didleur - vil-farfadet - michon - Dimitry.e -

1. Introduction générale (10) 4
1.1. Comprendre la FAQ (4) 5
1.2. Introduction au langage Perl (6) 7
2. Installation de Perl (4) 10
2.1. Comment savoir si Perl est installé sur ma machine ? (1) 11
2.2. Comment installer Perl ? (3) 12
3. S'initier à Perl (66) 14
3.1. Les commentaires (2) 15
3.2. Scalaires (12) 17
3.3. Listes (11) 25
3.4. Les listes associatives (9) 36
3.5. Structures de contrôle (20) 43
3.6. Les entrées/Sorties conversationnelles (4) 54
3.7. Expressions régulières (8) 57
3.7.1. Classes de caractères (7) 58
4. Perl avancé (45) 65
4.1. Les fichiers (11) 66
4.2. Les processus (5) 74
4.3. Les modules (14) 76
4.3.1. Installation des modules (3) 79
4.3.2. Exemple d'utilisation de quelques modules (2) 86
4.3.3. Mettre à jours ses Modules (3) 89
4.3.4. Modules CPAN intéressants (2) 91
4.4. Les références (9) 92
4.5. Opérateurs (3) 100
4.6. Pragmas (3) 104
5. Programmation orientée objets en Perl (24) 105
5.1. Introduction (8) 106
5.2. La vie d'un objet en Perl (5) 108
5.3. Méthodes et accès aux champs (5) 111
5.4. Utilisation de l'orienté objet en Perl (6) 113
6. Gestion des dates (3) 116
6.1. Gestion des dates (1) 117
6.2. Editeurs de texte utilisés par les perléens (1) 118
6.3. Je ne trouve pas mes réponses (1) 119
7. Codes sources utiles (62) 120
7.1. Des codes sources (61) 121
7.1.1. Bioinformatique (3) 122
7.1.2. Expressions régulières (3) 125
7.1.3. Fichiers et répertoires (11) 127
7.1.4. Réseaux (6) 144
7.1.5. Gestions des dates (7) 150
7.1.6. Gestions des tableaux (Array) (3) 156
7.1.7. Quelques unilignes perl (15) 161
7.1.7.1. Traitements de fichiers (11) 163
7.1.7.1.1. Insertion de lignes dans un fichier (6) 166
7.1.7.2. Web (1) 168
7.1.7.3. Divers (1) 169
7.1.8. Terminal (5) 170
7.1.9. Web (1) 179
7.1.10. Divers (6) 181
7.2. Téléchargements (1) 185
8. Divers (1) 186

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > Introduction générale

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > Introduction générale > Comprendre la FAQ

A qui s'adresse la FAQ ?

Auteurs : GLDavid ,

La FAQ Perl a pour but de vous aider dans votre apprentissage du langage Perl. Elle s'adresse en prime au débutant ainsi qu'à toute personne souhaitant s'initier à un langage de programmation. Mais elle convient aussi au programmeur averti qui aura besoin de toujours revoir quelques notions importantes de ce langage.

Comment participer à cette FAQ ?

Auteurs : Djibril ,

Cette FAQ est ouverte à toute collaboration. Pour éviter la multiplication des versions, il serait préférable que toute collaboration soit transmise aux administrateurs de la FAQ.

Plusieurs compétences sont actuellement recherchées pour améliorer cette FAQ :

1. **Rédacteur : bien évidemment, toute nouvelle question/réponse est la bienvenue ;**
2. **Correcteur : malgré nos efforts, des fautes d'orthographe ou de grammaire peuvent subsister. Merci de contacter les administrateurs si vous en débusquez une... Idem pour les liens erronés.**

Les autres sources d'information

Auteurs : GLDavid , 50Nio , Djibril ,

Mis à part cette FAQ ainsi que le forum Perl pour vos questions, les sites discutant de Perl ne manquent pas. Avant tout, le site de référence est le CPAN. Vous y trouverez absolument tout sur tout ou presque. Les modules sont des outils (globalement) propres et officiels, à utiliser sans crainte. C'est une véritable boite à outils pour tous les domaines informatiques professionnels comme pour le loisir.
Dès que vous vous posez une question du type :

- **Comment chercher si un module existe pour ce que je veux faire ?**
- **Comment faire un truc en Perl ?**
- **Quelqu'un a-t-il une toolbox pour faire ça ?**
- **Peut-on envoyer des mails via SNMP en Perl ou autrement ?**
- **Comment analyser facilement un fichier XML en Perl ?**
- **Puis-je gérer les cookies dans Firefox en Perl ?**
- **Comment créer un arbre en Perl ?**
- **...**

Ayez le réflexe !!
A noter aussi des sites de cours disponibles tels :

- **Bien débuter en Perl, par Sylvain Lhullier**
- **Les tutoriels et articles de notre rubrique Perl**

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

- Source **Des scripts à votre disposition**

lien : **Recherche CPAN**

Remerciements

Auteurs : Djibril ,

Nous tenons à remercier toute l'équipe de developpez.com (ced, stoyak...) pour la relecture de la FAQ, notamment :

- **ClaudeLELOUP pour sa très grande patience et son efficacité.**

Nous remercions également tous ceux qui nous font régulièrement des propositions, des corrections ou toutes idées d'améliorations.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Dois-je connaitre un langage de programmation ?

Auteurs : GLDavid ,

Comme expliqué auparavant, la FAQ s'adresse aussi bien au débutant qu'au professionnel ou à l'utilisateur expert de Perl. Aussi, pour le néophyte, il n'est pas indispensable d'avoir des connaissances préalables de quelque langage que ce soit. Perl peut aussi bien être votre premier langage de programmation.

Qu'est-ce que Perl ?

Auteurs : Djibril ,

Perl est un langage de programmation de haut niveau écrit par Larry Wall et un bon millier de développeurs avec un héritage éclectique. Il dérive de l'omniprésent langage C et, dans une moindre mesure, de Sed, Awk, du Shell Unix et d'au moins une douzaine d'autres langages et outils. Son aisance à manipuler les processus, les fichiers et le texte le rend particulièrement bien adapté aux tâches faisant intervenir le prototypage rapide, les utilitaires système, les outils logiciels, les gestionnaires de tâches, l'accès aux bases de données, la programmation graphique, les réseaux et la programmation web.
Ces points forts en font un langage particulièrement populaire auprès des administrateurs système et des auteurs de scripts CGI. Mais d'autres aussi l'utilisent : des mathématiciens, des généticiens, des bioinformaticiens, des journalistes et même des managers. A votre tour !!

Bref historique de Perl

Auteurs : Djibril ,

Perl a été créé en 1987 par un linguiste du nom de Larry Wall. Perl signifie *Practical Extraction and Report Language* : C'est un langage créé pour gérer facilement des fichiers et chaînes de caractères.
Petit à petit, Perl a été développé grâce à l'extension du web, notamment avec l'utilisation des news : comp.lang.perl.misc. Certains parlent même de Perl comme d'un langage *"Pathologically Eclectic Rubbish Lister"*, en gros une liste de tout et n'importe quoi puisque ce langage reprend beaucoup de fonctionnalités utiles. C'est un peu le tout-en-un !

Avantages/Inconvénients de Perl

Auteurs : Djibril ,

Avantages

- **pour toutes les tâches liées à la manipulation de chaînes de caractères ;**
- **temps de développement beaucoup plus court qu'en JAVA, C ou C++, par exemple ;**
- **robustesse : pas d'allocation mémoire à manipuler, la gestion de mémoire est prise en charge par Perl ;**
- **pas de limitation sur la taille des données ou de leur contenu ;**
- **relativement facile à apprendre, communauté ouverte et dynamique ;**
- **regroupement des modules Perl et de leur documentation sur le site en CPAN ;**
- **documentation très abondante ;**
- **portabilité ;**
- **gratuité et beaucoup de programmes disponibles sur Internet ;**
- **dernières normes informatiques intégrées (comme la programmation objet).**

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Inconvénients

- **il est moins adapté pour le calcul scientifique, mais il est possible de créer des extensions et faire appel à des fonctions d'une librairie C ou C++ ;**
- **sa permissivité peut rendre difficile la portabilité ou la réutilisation du code : elles dépendent de la rigueur du programmeur.**

Les outils Perl

Auteurs : Djibril ,

Voici une liste de lignes de commande vous permettant d'obtenir de la documentation sur Perl, une sorte de manuel en ligne découpé en sections :

- **man perl ou perldoc perl : généralités sur la doc, mots-clefs...**
- **man perlsyn ou perldoc perlsyn : Syntaxe ;**
- **man perlfunc ou perldoc perlfunc : Fonctions intégrées ;**
- **man perlvar ou perldoc perlvar : Variables prédéfinies ;**
- **man perlobj ou perldoc perlobj : Objets ;**
- **man perlfaq ou perldoc perlfaq : FAQs (classées par thèmes) ;**
- **man perldebug ou perldoc perldebug : Mise au point ;**
- **man perlpragma ou perldoc perlpragma : informations sur les pragmas (n'existe que pour perl supérieur ou égal à perl 5.10 ;**
- **man perlreguts ou perldoc perlreguts : informations sur les internals du moteur des expressions régulières (n'existe que pour perl supérieur ou égal à perl 5.10 ;**
- **man perlunitut ou perldoc perlunitut : introduction à Unicode dans Perl (n'existe que pour perl supérieur ou égal à perl 5.10.**

Mon premier programme Perl

Auteurs : Djibril ,

Toutes les versions de Perl

```
#!/usr/bin/perl
use strict;
use warnings;

print "Bienvenue dans le monde de Perl\n";
```

La première ligne de notre programme indique le chemin vers l'interpréteur de Perl. Sous Windows, vous pouvez laisser ce chemin car Perl s'y retrouve de toute manière !

- **use : permet de charger un module Perl ;**
- **strict et warnings : sont deux modules internes à Perl permettant de vérifier la syntaxe de votre code et de vous alerter à la compilation du programme ;**
- **print : permet d'afficher sur la sortie standard STDOUT (sur la console). \n permet de passer à la ligne (retour à la ligne).**

Pour ceux disposant d'une version de Perl supérieure ou égale à 5.10, vous pouvez utiliser à la place de print le mot say qui met automatiquement un retour chariot (comme *println* en JAVA). Mais pour pouvoir utiliser cette fonctionnalité, il faut faire appel à Perl 5.10 dans ses programmes et le code minimal ressemble à :

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

version supérieure ou égale à 5.10

```
#!/usr/bin/perl
use strict;
use warnings;
use 5.10.0;

say "Bienvenue dans le monde de Perl";
```

ou faire appel au module interne de perl feature de la sorte :

version supérieure ou égale à 5.10

```
#!/usr/bin/perl
use strict;
use warnings;
use feature 'say';

say "Bienvenue dans le monde de Perl";
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > Installation de Perl

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Tout système d'exploitation

Auteurs : Djibril ,

Quel que soit votre système d'exploitation (Linux/Unix, Windows, Macintosh...), il vous suffit d'ouvrir un terminal et de taper la commande suivante :

```
perl -v
```

Si vous avez un message d'erreur spécifiant qu'il ne reconnaît pas Perl alors ce dernier n'est pas installé.

Sous DOS - Perl non installé

```
C:\>perl -v
perl n'est pas reconnu en tant que commande interne ou externe, un programme exécutable ou un
 fichier de commande
```

Ou

Sous DOS - Perl installé

```
C:\>perl -v

This is perl, v5.10.1 built for MSWin32-x86-multi-thread
(with 4 registered patches, see perl -V for more detail)

Copyright 1987-2009, Larry Wall

Binary build 1008 [294165] provided by ActiveState http://www.ActiveState.com
Built Dec  9 2010 06:00:35

Perl may be copied only under the terms of either the Artistic License or the
GNU General Public License, which may be found in the Perl 5 source kit.

Complete documentation for Perl, including FAQ lists, should be found on
this system using "man perl" or "perldoc perl".  If you have access to the
Internet, point your browser at http://www.perl.org/, the Perl Home Page.
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sur Windows

Auteurs : Djibril ,

Sous Windows, Perl n'est pas installé par défaut. Vous devez installer le package disponible gratuitement sur le site d'ActiveState (ActivePerl), c'est la distribution standard.
A ce jour (15/06/2011), il y a deux versions récentes de Perl disponible en téléchargement :

1. **ActivePerl 5.14.1.1401 ;**
2. **ActivePerl 5.12.4.1205.**

D'un point de vue personnel, je recommande d'installer la version 5.12 car on a plus de facilités pour trouver des modules CPAN compatibles (Sous Windows via ppm).
N.B : Il est maintenant impossible de récupérer les anciennes versions de Perl (5.10.1.1008, 5.8.9.829). ActiverPerl ne met à disposition que les deux versions citées ci-dessus.
Vous pouvez débattre de cette politique d'ActiveState sur le forum : .

Sachez qu'il existe également Strawberry Perl à la place d'ActivePerl, mais cette FAQ ne repose que sur ActivePerl pour son installation et celle des modules CPAN.

lien : ActiveState
lien : ActivePerl
lien : Strawberry Perl

Sur Linux

Auteurs : GLDavid , Djibril ,

Sur les principales distributions Linux, Perl est déjà installé.
Autrement, pour les fervents adeptes de la compilation, vous pourrez aussi installer les sources de Perl à partir du site officiel.
Si vous souhaitez installer une version d'ActivePerl sous Linux, c'est également possible. Il existe des packages zip, des fichiers rpm ou même des packages Debian disponibles. Tout se trouve ici. L'installation est très simple. Par exemple sous Debian :
- téléchargez le package Debian
- installation

```
dpkg -i ActivePerl-5.12.4.1205-i686-linux.deb
```

ActivePerl sera installé dans le répertoire /opt/ActivePerl-5.12

lien : Installer ActivePerl 5.8 sous Linux

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

lien : **Installer ActivePerl 5.10 sous Linux**

Sur Macintosh

Auteurs : Stoyak ,

Je suppose que vous disposez d'un ordinateur suffisamment récent, et que vous êtes donc sous Mac OS X !! *Petite note spéciale pour tous les développeurs sous Mac, je vous conseille une petite visite sur le projet Omega, la visite est très instructive !*

1) Installation de Perl par défaut :

Eh bien, bonne nouvelle, Perl est installé par défaut, version 5.6.

Vérifiez par vous-même :

```
perl -v
```

S'il est bien présent, vous verrez la version que vous utilisez !

Vous ouvrez alors une fenêtre Terminal, et appelez votre script de la manière suivante :

```
perl script.pl
```

2) Installer Perl, version ActiveState :

Si comme moi, vous venez de l'univers Windows, que vous avez l'habitude d'utiliser ActiveState pour Perl, bonne nouvelle ! Il est aujourd'hui disponible sous Mac OS X !!

Quel est l'intérêt, me direz-vous, de le télécharger alors qu'il est présent par défaut ? Eh bien, une version plus récente, le " Perl Package Manager " (PPM) pour faciliter l'installation des modules (très pratique), une aide en ligne...

ActivePerl est distribué pour Mac OS X sous la forme d'un disque image (.dmg file), à télécharger. Il contient l' " Installer Package " (.pkg bundle), sur lequel il faut double-cliquer pour démarrer l'installation !

ActivePerl est alors installé dans le répertoire /usr/local/ActivePerl-5.12.

Vous devez ensuite ajouter *:/usr/local/ActivePerl-5.12/bin* dans votre variable d'environnement pour pouvoir utiliser PPM et exécuter vos scripts avec ActivePerl.

Pour connaître votre interpréteur de commande, tapez dans votre Terminal :

```
echo $SHELL
```

Par exemple, si vous êtes sous Bash, vous obtiendrez :

```
/usr/local/bin/bash
```

Ainsi, pour un Shell Tcsh, ajoutez dans votre fichier .cshrc

```
setenv PATH /usr/local/ActivePerl-5.12/bin:$PATH
```

pour un Shell bash, ajoutez dans votre fichier .profile

```
export PATH=/usr/local/ActivePerl-5.12/bin:$PATH
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > S'initier à Perl

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > S'initier à Perl > Les commentaires

Comment commenter une ligne de code ?

Auteurs : Djibril ,

En Perl, comme dans tous les langages de programmation, il est recommandé d'insérer des commentaires pour une meilleure clarté. Le commentaire commence à partir d'un caractère dièse (#) s'étend jusqu'à la fin de la ligne (retour chariot). Mais, cette ligne ne commence pas nécessairement par dièse comme le montre l'exemple ci-dessous.

```
#!/usr/bin/perl
use strict;
use warnings;

# Voici une ligne de commentaire.
print "Bonjour\n"; # Affichage du mot bonjour (autre commentaire).
```

Comment commenter des portions de code plus larges ?

Auteurs : 2Eurocents , Philou67430 , Djibril ,

Il est vrai que le mécanisme de commentaires commençant par un dièse (#) et se terminant à la fin de la ligne manque de souplesse, notamment dans le cas où l'on souhaite commenter de larges blocs de code. Dans ce cas, il est possible de détourner un mécanisme avancé de Perl qui est conçu pour permettre d'avoir des scripts contenant leur propre documentation : POD (Plain Old Documentation - Bonne Vieille Documentation - très utile par ailleurs).
Les directives POD =for et begin/=end permettent de créer facilement de larges blocs de texte qui seront ignorés par le compilateur et qui ne produiront pas non plus d'effet visible lorsque le fichier sera traité par un formateur POD. Ces instructions constituent donc un moyen très pratique d'insérer des portions de documentation interne dans vos sources.
La directive =for est identique à une paire begin/=end, la seule différence étant que la documentation ne concerne qu'un paragraphe, dont la fin est matérialisée par une ligne vide. Il est obligatoire d'ajouter la directive =cut pour indiquer au compilateur que la documentation est terminée et qu'il doit reprendre l'analyse de la source Perl.
Évitez la forme =begin/=end, sauf dans le cas où les explications sont abondantes et sur plusieurs paragraphes ou bien lorsque du code est inséré dans ces explications.

```
$i=5; # ici, ce code est interprété
=for mon commentaire:
blablabla
blablabla
blablabla

=cut
# mon code
#...

=for autre commentaire:
blablabla
blablabla
blablabla

=cut

=begin test:
blablabla
blablabla

blablabla
blablabla

for (1..12) {
  say $_; # plus adequate
}
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
=end test

=cut

print $i."\n"; # mais l'interprétation reprend ses droits, ici même.
```

Pour en savoir plus sur POD :

- **perldoc pod (sous Windows) ;**
- **Documentez vos modules Perl avec Pod.**

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > S'initier à Perl > Scalaires

Qu'est-ce qu'un scalaire ?

Auteurs : 2Eurocents ,

Une variable ou un scalaire (traduction impropre et systématique de l'anglais 'scalar' qui signifie "variable" ou "quantité variable") est le plus petit élément d'information que puisse traiter Perl. En Perl, un scalaire se repère au premier caractère de son identifiant (de son nom). Ce premier caractère est obligatoirement le symbole "dollar" ($). Le nom d'un scalaire peut être composé de chiffres, de lettres et de certains caractères spéciaux (_), mais il ne doit jamais commencer par un chiffre. On a coutume de dire qu'il s'agit d'un élément atomique, au sens où il n'est pas possible simplement de le diviser en éléments plus petits (du grec atomos qui signifie indivisible - "tomos" pour couper avec le préfixe privatif "a").

Que peut contenir un scalaire ?

Auteurs : 2Eurocents ,

Une variable scalaire peut contenir à peu près n'importe quoi, Perl étant un langage dit faiblement typé. Les langages faiblement typés ne font pas la différence, *a priori*, entre nombres entiers, nombres à virgule flottante, caractères et chaînes de caractères. Cela ne pose cependant pas de problème en Perl, ainsi qu'on le verra plus loin...

Comment définir une variable scalaire ?

Auteurs : 2Eurocents ,

Il suffit de l'utiliser, c'est-à-dire d'y faire référence en utilisant son identifiant (nom complet, avec le $ initial). Contrairement à de nombreux autres langages de programmation, Perl n'impose pas de déclaration préalable des variables. Ainsi, toute utilisation de variable crée automatiquement celle-ci. Pour pallier les nombreux risques que cette absence de déclaration des variables peut faire peser sur le développement, il existe des options permettant d'imposer une syntaxe stricte avec déclaration préalable à tout usage. Ainsi, pour plus de sécurité, il est préférable d'écrire, au début du script :

```
use strict;
```

On parle alors d'utiliser les "structures" - si quelqu'un s'oppose à ce *barbarisme*, qu'il parle maintenant ou se taise à jamais !!

Comment déclarer une variable scalaire ?

Auteurs : 2Eurocents ,

Les mots-clefs de déclaration de variables sont my et our.
Ces mots-clefs sont valables pour tous types de variable, et pas uniquement pour les variables scalaires. La différence entre my et our est abordée plus loin dans ce document. Dans l'immédiat, l'usage de my conviendra pour la plupart des cas. Ainsi, pour déclarer une variable $i, on écrira simplement :

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
my $i;
```

Comment affecter une valeur à une variable scalaire ?

Auteurs : 2Eurocents ,

L'opérateur d'affectation est tout simplement l'opérateur =. Ainsi, il est possible d'écrire :

```
my $i=0;
```

pour affecter une valeur initiale à la variable, dès son initialisation. Par la suite, on peut écrire :

```
$i = 5;

# voire :
my $j = $i;

# ou bien :
$i ='a';
$i ='test de chaîne';
$i =3.1415;
```

A l'issue de ces trois dernières affectations, $i vaut le caractère "a", pour la première, la chaîne "test de chaîne" pour la deuxième et 3.1415 (le nombre à virgule flottante) pour la troisième.

Les nombres peuvent être exprimés en

1. **base décimale ($i = 100;) ;**
2. **octal (il suffit de commencer la valeur par un zéro : $i = 0144;) ;**
3. **hexadécimal (en commençant la valeur par un zéro suivi par un x : $i = 0x64;).**

Et si je n'affecte pas de valeurs à mes variables ?

Auteurs : 2Eurocents ,

Les variables auxquelles aucune valeur n'a été affectée sont initialisées automatiquement par l'interpréteur Perl. La valeur implicite d'initialisation est une valeur spécifique un peu magique nommée "undef".

- **dans des opérations arithmétiques, elle vaut zéro ;**
- **dans les opérations de chaînes de caractères, elle équivaut à la chaîne vide, mais attention, toutefois car son usage provoque des affichages de messages d'avertissement si les "warnings" sont actifs (use warnings;) ;**
- **il ne faut pas la tester de manière classique avec les opérateurs de comparaison (voir plus loin), mais au moyen d'une fonction spécifique : defined() qui retourne vrai si la variable est définie et faux si elle vaut undef ;**
- **il est possible de l'affecter soi-même à une variable déjà définie, soit par une affectation $variable=undef; soit par une fonction undef($variable).**

Quelles opérations peut-on réaliser sur ces variables ?

Auteurs : 2Eurocents , Djibril ,

Les principaux opérateurs arithmétiques sont disponibles. Ainsi, il est possible d'utiliser :

- **+ pour l'addition ($a = $b + 75;) ;**

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

- **- pour la soustraction (l'opérateur - existe aussi en version "unaire", pour signifier un nombre négatif) ($a = -75 -$b;) ;**
- *** pour la multiplication ($a = $b * $c;) ;**
- **/ pour la division ($a = 83 / 17;) ;**
- **% pour le modulo (reste de la division entière) ($b = 83 % 17;) ;**
- **** pour l'exponentiation (l'élévation à une puissance, comme $a = $b ** 3;) ;**
- **++ est un opérateur (unaire) de pré- ou post incrémentation de la variable sur laquelle il porte (++$a ou $a++) ;**
- **-- est un opérateur (unaire) de pré- ou post décrémentation de la variable sur laquelle il porte (--$a ou $a--).**

Il existe aussi des opérateurs spécifiques aux traitements sur les chaînes de caractères :

- **. (le point) est l'opérateur de concaténation de chaînes ($a = "abcd" . "efgh";) ;**
- **x est l'opérateur de répétition qui répète la chaîne le précédant autant de fois qu'indiqué par le nombre suivant ($a = "*" x 40; pour obtenir quarante astérisques) ;**
- **= est l'opérateur de mise en correspondance avec une expression rationnelle (gros chapitre du langage qui sera abordé ultérieurement).**

Certains opérateurs sont qualifiés de "*binaires bit à bit*" :

- **~ pour la négation bit à bit ($i = ~ $j;) ;**
- **<< est un opérateur de décalage bit à bit vers la gauche ($a = $b << 3;) ;**
- **>> est un opérateur de décalage bit à bit vers la droite ($a = $b >> 3;) ;**
- **& effectue un ET bit à bit entre ses opérandes ($zero = 0x36 & 0xC9;) ;**
- **| pour un OU bit à bit ($deux_cent_cinquante_cinq = 0x36 | 0xC9;) ;**
- **^ pour un OU EXCLUSIF entre deux valeurs.**

Il existe aussi des opérateurs de test, de comparaison, et ces opérateurs existent sous deux formes, selon que l'on souhaite comparer des nombres entiers ou bien des chaînes de caractères :

- **== ou eq pour tester l'égalité numérique ou alphabétique ;**
- **< ou lt pour l'infériorité stricte ;**
- **<= ou le pour l'infériorité ou égalité ;**
- **> ou gt pour la supériorité stricte ;**
- **=> ou ge pour la supériorité ou égalité;**
- **!= ou ne pour la différence ;**
- **<=> ou cmp sont des opérateurs spéciaux de comparaison, qui retournent -1, 0 et 1 selon lequel de leurs opérandes est le plus grand.**

```perl
#!/usr/bin/perl
use warnings;
use strict;

my $nombre1 = 12;
my $nombre2 = 22;

my $comparaison1 = 12 <=> 21;
my $comparaison2 = 21 <=> 12;
my $comparaison3 = 12 <=> 12;
my $comparaison4 = 'az' cmp 'zz';
my $comparaison5 = 'zz' cmp 'az';
my $comparaison6 = 'zz' cmp 'zz';

print "$comparaison1\n";    # -1
print "$comparaison2\n";    # 1
print "$comparaison3\n";    # 0
print "$comparaison4\n";    # -1
print "$comparaison5\n";    # 1
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
print "$comparaison6\n";    # 0

if ( 1 < 10 ) {
  print "< : 1 est inférieur à 10\n";
}

if ( 1 lt 10 ) {
  print "lt : 1 est inférieur à 10\n";
}
```

Et enfin, des opérateurs logiques complètent la panoplie :

- **&& ou and permettent de réaliser la conjonction entre deux conditions (le ET logique) ;**
- **|| ou or permettent de réaliser la disjonction entre deux conditions (le OU logique) ;**
- **xor permet de réaliser une disjonction exclusive (le XOR logique) ;**
- **! ou not permettent de réaliser la négation d'une condition.**

La différence entre les opérateurs logiques similaires vient, entre autres, des différences de priorité. Ils ne sont cependant pas toujours utilisés dans le même contexte. Notamment, l'usage des opérateurs logiques dans le style C (&&, ||) avec des listes (sujet abordé plus loin) risque de poser problème. Dans ce cas, l'usage des opérateurs en toutes lettres est plus sûr. Il existe encore d'autres opérateurs, dont les plus utiles sont :

- **?: qui permet de réaliser un choix entre deux valeurs, selon le test d'une condition ($i = ($a == $b)?$a:$b; $i vaudra $a si $a et $b ont la même valeur, $b s'ils sont différents) ;**
- **, qui procède à l'évaluation de son membre de droite, puis à celle du membre de gauche, avant de garder le résultat de la dernière évaluation (ça parait compliqué et inutile, comme ça, mais quand on maîtrise bien, ça peut réellement être très puissant) ($i = ($j=100),($k=50); fait que $i et $j valent 100, $k vaut 50).**

Et il existe aussi des formes abrégées des principaux opérateurs réfléchis (même variable présente des deux côtés de l'affectation). Ces opérateurs posent les mêmes difficultés de maintenance ou de compréhension du code qu'en C. On trouve : +=, -=, *=, /=, .=, <<=, etc.

Opérateur defined-or (depuis Perl 5.10):
Cet opérateur est //, à ne pas confondre avec ||. $x // $y équivaut à (defined $x ? $x : $y).
// vérifie si la variable $x est définie, si c'est le cas, il retourne $x sinon retourne $y.
$z //= $x; équivaut à if (not defined $z) { $z = $x; }
Voici un exemple de code

```
#!/usr/bin/perl
use warnings;
use strict;

my $x = 10;
my $y = 20;
my $z;
my $age = $x // $y;
print "Mon age : $age ans\n";

$age = $z // $y;
print "Mon age : $age ans\n";

$z //= $x; # équivaut à if ( not defined $z ) { $z = $x; }
print "\$z : $z\n";
```

```
Mon age : 10 ans
Mon age : 20 ans
$z : 10
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Qu'elle est la différence par rapport à || et à quel moment, il est préférable d'utiliser // ?
Prenons l'exemple d'une procédure qui retourne une valeur. L'exemple suffira à la compréhension de l'opérateur.

```
#!/usr/bin/perl
use warnings;
use strict;

sub doubler {
  my $valeur = shift || 10;
  my $double = ($valeur * 2);
  print "$double\n";
}
sub doubler_version2 {
  my $valeur = shift // 10;
  my $double = ($valeur * 2);
  print "$double\n";
}

doubler(17); # imprime 34
doubler();   # imprime 20
doubler(0);  # imprime 20 <= Attention, piège
print "============\n";
doubler_version2(17); # imprime 34
doubler_version2();   # imprime 20
doubler_version2(0);  # imprime 0 <= c'est mieux
```

Mais comment faire la différence entre entiers, flottants, chaînes, etc. ?

Auteurs : 2Eurocents ,

Perl sait très bien faire la différence tout seul.
En fait, en fonction des opérateurs utilisés, Perl détermine automatiquement la façon dont il faut traiter le contenu des variables. C'est la notion de *"contexte"*. Il existe deux contextes principaux : contexte scalaire et contexte de list.
Le contexte scalaire, lui, possède des nuances : contexte scalaire *numérique*, contexte scalaire de *chaîne* et contexte scalaire *booléen*.
Lorsque l'on utilise des opérateurs numériques ou des fonctions numériques (que l'on verra plus tard), Perl modifie automatiquement le contexte pour traiter les variables comme des nombres. Qui plus est, si l'un de ces nombres contient un . (le séparateur décimal), par exemple, tous les nombres en jeu dans l'opération sont automatiquement promus comme nombres flottants.
Attention toutefois, car l'évaluation de chaînes de caractères en contexte numérique peut provoquer des résultats inattendus. Ainsi, l'évaluation suivante peut tout à fait donner un résultat, qui s'il est cohérent du point de vue de Perl n'en est pas moins faux :

```
$i = "2eurocents" + "Code 100 sous 6"; print $i."\n";
```

produira 2, car "2eurocents" est évalué à 2 en contexte numérique et "code 100 sous 6" n'arrive pas à être évalué (le premier mot n'est absolument pas numérique).

```
$i = "2eurocents" + "100sous6"; print $i."\n";
```

produira 102, "2eurocents" est toujours évalué à 2 et "100sous6" est évalué à 100, c'est-à-dire à toute la portion numérique initiale de la chaîne.
Heureusement, l'interpréteur Perl peut nous aider par des messages d'avertissements lors de telles opérations, si la directive *warnings* a été utilisée (paramètre -w du shebang ou clause use warnings;). Cela peut être un avantage, comme un inconvénient. Il importe que le programmeur teste au préalable les cas où cela peut lui poser des problèmes de cohérence car le langage ne le fera pas pour lui. Ainsi, en Perl, il est possible de faire aussi bien :

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
$volume = "10 l" + "5 dm³";
```

qui est cohérent, que :

```
$melange = "3 choux" + "18 paires de carottes";
```

qui n'a absolument aucun sens, pour le plus grand bonheur pédagogique de toute une génération d'instituteurs. Heureusement, le programmeur sait ce qu'il fait, la plupart du temps... le vrai danger vient le l'utilisateur, tout le monde le sait ! En fait, ces préoccupations valent surtout pour les variables sur lesquelles l'utilisateur peut avoir le contrôle (zones saisies, données provenant de fichiers, etc.).

Que faire avec les valeurs numériques ?

Auteurs : 2Eurocents ,

Bien que Perl ne soit pas réellement performant dans le domaine du calcul numérique (on verra pourquoi d'ici peu), il est tout à fait possible de mener quelques opérations. Les opérateurs arithmétiques sont disponibles, on l'a vu, mais il y a aussi quelques fonctions :

- **abs() pour les valeurs absolues ;**
- **atan2() pour la tangente inverse, ainsi que cos() et sin() pour les amoureux de la trigonométrie ;**
- **log() et exp() pour le logarithme népérien et les puissances de e ;**
- **sqrt() pour les racines carrées ;**
- **int() pour tirer les parties entières ;**
- **hex() ou oct() pour des changements de base numérique ;**
- **rand() et srand() permettent d'utiliser et d'initialiser le générateur de valeurs pseudoaléatoires.**

Et les chaînes de caractères ?

Auteurs : 2Eurocents ,

Quoi, les chaînes de caractères ? En fait, en Perl, les chaînes de caractères sont des scalaires comme les valeurs numériques. Pour être parfaitement rigoureux, on devrait même dire que les valeurs numériques sont des scalaires comme les chaînes de caractères !
En effet, Perl stocke toutes ses variables sous la forme de chaînes, ainsi que les programmeurs expérimentés peuvent s'en apercevoir en manipulant les fonctions pack/unpack... Perl ne fait même pas la différence entre caractères imprimables ou valeurs binaires : tout cela constitue des valeurs scalaires valides.

Comment exprimer une chaîne de caractères ?

Auteurs : 2Eurocents ,

Une chaîne de caractères est un scalaire dont la valeur est exprimée entre quotes. Nos claviers étant généreusement dotés de simple comme de double quote, nous avons le choix de l'expression.
Lorsque la chaîne de caractères est exprimée entre *doubles quotes*, l'interpréteur Perl poursuit son travail à l'intérieur et y interprète ce qu'il y rencontre. Il est ainsi possible d'y insérer des éléments tels que d'autres scalaires (préfixés par un $), des caractères spéciaux exprimés en notation "*back slash*" (tels que \n pour le saut de ligne, \r pour le retour de chariot, \t pour la tabulation...), etc.

```
$monde = "world !\n";
$politesse = "Hello $monde";
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
print $politesse;
```

Dans le cas où l'on souhaiterait accoler un scalaire et du texte fixe dans une chaîne de caractères, sans espace intermédiaire, il est nécessaire d'exprimer le scalaire différemment afin d'aider Perl à trouver les limites de son nom. On met ainsi le nom du scalaire entre accolades :

```
$racine = "programme";
$_1e_pers_sing = "Je ${racine}e";
$_2e_pers_sing = "Tu ${racine}es";
$_3e_pers_sing = "Il ${racine}e";
$_1e_pers_plur = "Nous ${racine}ons";
$_2e_pers_plur = "Vous ${racine}ez";
$_3e_pers_plur = "Ils ${racine}ent";
```

Il est possible de bloquer le travail d'interprétation de Perl en protégeant certains caractères par une barre oblique inversée. Ainsi, pour insérer des doubles quotes, un dollar, une arobas ou une barre inversée, il faut exprimer respectivement \", \$, \@ et \\.
De la même manière, pour exprimer certains caractères spéciaux dont on connait le code ASCII, il est possible d'exprimer ce dernier numériquement, à la suite d'un antislash : \0 (caractère NULL) ou \7 (caractère BELL) par exemple. Lorsque la chaîne est exprimée entre simples quotes, l'interpréteur Perl la prend telle quelle, sans examiner son contenu. Seuls les caractères ' et \ ont besoin d'y être protégés (' doit être écrit \', pour ne pas être confondu avec la fin de chaîne, et \ doit être protégé en \\ puisqu'il peut avoir un rôle spécial dans la protection de ').
Ainsi, "col1\tcol2\tcol3\n" représente un texte de trois colonnes séparées par des tabulations et terminé par un saut de ligne.
'col1\tcol2\tcol3\n' représente ce texte tel quel, avec les \, les t qui les suivent et le n final.

Mais que peut-on faire avec les chaînes de caractères ?

Auteurs : 2Eurocents ,

La vraie puissance de Perl, on le dit souvent, réside dans le traitement des chaînes de caractères. C'est particulièrement vrai dans l'application des expressions rationnelles, qui seront vues ultérieurement. Hors ce gros morceau du langage, de nombreux traitements sont possibles, que ce soit par le biais des opérateurs ou par des fonctions spécifiques :

- **l'opérateur de concaténation : "Bonjour "."le monde" ;**
- **l'opérateur de multiplication : "Bonjour " x 3 ;**
- **chop() supprime et renvoie le dernier caractère d'une chaîne de caractères ;**
- **chomp() supprime le dernier caractère d'une chaîne, uniquement s'il s'agit d'une fin de ligne (mais ne renvoie pas celui-ci !) ;**
- **reverse() retourne une chaîne de caractères (aimez-vous les palindromes ?) ;**
- **uc(), lc(), ucfirst() et lcfirst() sont des fonctions de changement de casse des chaînes passées en paramètre. uc() passe la chaîne en majuscules (upper case) et lc() en minuscules. Les variantes *first ne traitent que la première lettre ;**
- **oct() et hex() font des conversions de bases numériques. Ah, mais on en a déjà parlé quand on évoquait des fonctions numériques... Oui, mais en Perl, tout est chaîne de caractères, donc en fait, les fonctions numériques travaillent sur des chaînes qu'elles convertissent pour le traitement ;**
- **chr() et ord() font des conversions code ASCII/caractère. Ainsi, chr(65)="A" et ord("A")=65 ;**
- **length(), index(), rindex() et substr() sont des fonctions classiques de traitement de chaînes.**
  La fonction length() renvoie la longueur de la chaîne.
  Les fonctions index() et rindex() renvoient la position de la première ou de la dernière occurrence d'une sous-chaîne dans une chaîne, avec éventuellement une position de début de recherche. substr() retourne une portion d'une chaîne donnée, à partir d'une position fixée et d'une longueur précisée.
  Cette dernière fonction, substr() est intéressante à plus d'un titre car elle peut aussi servir à faire un remplacement de portion de chaîne, éventuellement de longueur variable.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://perl.developpez.com/faq/perl/

Elle est utilisée, dans ce cas, comme réceptacle d'une affectation : my $chaîne="Bonjour tout le monde"; substr ($chaîne, 3, 4) = " appétit"; remplace "jour" (position 3, longueur 4 dans la chaîne) par " appétit ". La chaîne est agrandie et tout est pour le mieux ;

- **pack() et unpack() sont des fonctions très avancées qui permettent toutes sortes de codages et décodages de valeurs binaires, décimales, hexadécimales, tant sur des nombres que sur des chaînes. Elles sont très utilisées dès que vous avez à coder/décoder des valeurs binaires provenant de dialogues directs avec différents matériels, protocoles, systèmes...**
- **print(), printf() et sprintf() sont, bien sûr, des fonctions de dialogue. Elles permettent de dialoguer aussi bien avec la console, qu'avec des fichiers. La fonction print affiche simplement ses paramètres, là où les fonctions printf et sprintf permettent des formatages complexes, comme leurs homologues en C dont elles sont issues ;**
- **tr/// (ou y///) est une fonction spéciale permettant de transformer, dans une chaîne, tous les caractères d'un ensemble en caractères d'un second ensemble. Elle est apparentée aux expressions rationnelles.**

D'autres fonctions sont disponibles, pour traiter toutes sortes de scalaires, et lorsque ce n'est pas suffisant, le langage peut être étendu par l'usage de modules, ainsi qu'on le verra par la suite.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > S'initier à Perl > Listes

Je n'ai pas qu'une valeur à traiter, comment fais-je ?

Auteurs : 2Eurocents ,

Nous avons vu que Perl gérait des variables scalaires, c'est-à-dire atomiques. Il est aussi capable de gérer des types de variables un peu plus complexes, qu'il sera possible de décomposer en scalaires. Ces types "agrégés" sont principalement au nombre de deux :

- **les listes simples, que l'on rapprochera de la notion de table et qui font l'objet de ce chapitre ;**
- **les listes associatives, ou tables associatives, qui feront l'objet du chapitre suivant.**

Qu'est-ce qu'une liste simple ?

Auteurs : 2Eurocents , Djibril ,

Une liste simple ou ordinaire est un type de variable particulier contenant des scalaires.
Une liste s'exprime par une ou plusieurs valeurs séparées par des virgules et mises entre parenthèses :

```
my @list = (1, $toto, 'Hello World', "Histoire de $toto", 3.1415);
```

C'est l'expression d'une liste rassemblant la valeur entière 1, le contenu du scalaire $toto, la chaîne 'Hello World', la chaîne constituée par la jonction entre "Histoire de " et le contenu du scalaire $toto et pour finir, la valeur flottante 3.1415.
La nature des scalaires contenus dans une liste n'a absolument aucune espèce d'importance pour Perl comme on vient de le voir, mais fonctionnellement le programmeur a peut-être intérêt à veiller à la consistance des données.
Il est possible de constituer les listes en utilisant quelques opérateurs spécifiques. L'opérateur d'intervalle (que l'on n'a pas encore abordé) ".." crée explicitement une liste commençant à la valeur qui le précède et se termine à la valeur qui le suit (sous réserve d'une certaine cohérence de type).

Opérateur .. :
- **(5..15) crée une liste de tous les entiers de 5 à 15 ;**
- **(-5..5) crée une liste des entiers de -5 à 5 ;**
- **('a'..'f') crée une liste de tous les caractères de 'a' à 'f' ;**
- **('A'..'F') crée une liste de tous les caractères de 'A' à 'F' (distincte de la précédente - attention à la casse !) ;**
- **('a'..'F') crée une liste des caractères de 'a' à 'z' et (A..f) crée une liste des caractères de 'A' à 'Z'. En effet, la génération de liste part de la première valeur, et monte jusqu'à essayer de rencontrer la dernière. Malheureusement, elle bute en cours de route sur la limite de cohérence de l'ensemble ('z' pour les minuscules et 'Z' pour les majuscules) et elle s'y arrête ;**
- **('aaaa'..'zzzz') crée une liste de toutes les combinaisons de 4 lettres minuscules (déjà beaucoup... qui a dit "attaque brute-force ?).**

```
#!/usr/bin/perl
use warnings;
use strict;

my @exemple1 = ( 5 .. 15 );
my @exemple2 = ( -5 .. 5 );
my @exemple3 = ( 'a' .. 'f' );
my @exemple4 = ( 'A' .. 'F' );
my @exemple5 = ( 'a' .. 'F' );
my @exemple6 = ( 'aa' .. 'dd' );
print "@exemple1\n";    # 5 6 7 8 9 10 11 12 13 14 15
print "@exemple2\n";    # -5 -4 -3 -2 -1 0 1 2 3 4 5
print "@exemple3\n";    # a b c d e f
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
print "@exemple4\n";    # A B C D E F
print "@exemple5\n";    # a b c d e f g h i j k l m n o p q r s t u v w x y z
print "@exemple6\n";    # a b c d e f g h i j k l m n o p q r s t u v w x y z
# aa ab ac ad ae af ag ah ai aj ak al am an ao ap aq ar as at au av aw ax ay az ba bb bc bd be bf bg bh
# bi bj bk bl bm bn bo bp bq br bs bt bu bv bw bx by bz ca cb cc cd ce cf cg ch ci cj ck cl cm cn co cp c
# cr cs ct cu cv cw cx cy cz da db dc dd
```

L'opérateur de multiplication x, impose quelques précautions de parenthèses :

- **('A' x 3, 'B' x 2, 'C') crée une liste constituée de 'AAA', 'BB' et 'C'.**
- **(('A') x 3, ('B') x 2, 'C') crée une liste constituée de 'A', 'A', 'A', 'B', 'B' et 'C'. Les parenthèses introduites créent des listes d'un seul élément ('A' ou 'B') qui sont elles-mêmes multipliées ou dupliquées. Les listes étant des éléments entre parenthèses, il est possible d'introduire des listes dans des listes. Les listes qui sont ainsi insérées sont alors "aplaties" dans la liste réceptacle. Cela signifie que l'on n'introduit pas la liste en tant que telle, mais plutôt son contenu.**

```
#!/usr/bin/perl
use warnings;
use strict;

my @exemple1 = ( 'A' x 3, 'B' x 2, 'C' );
my @exemple2 = ( ('A') x 3, ('B') x 2, 'C' );
print "@exemple1\n";    # AAA BB C
print "@exemple2\n";    # A A A B B C
```

Cette notion d'aplatissement des listes les unes dans les autres est un concept essentiel dans le traitement des ensembles de valeurs par Perl : (1, 2, (3, 4, (5, 6), 7), 8) est une liste dans laquelle on introduit une sous-liste qui contient elle-même une sous-liste. Au final, on obtient une liste "plate", c'est-à-dire sans niveaux d'imbrication, de 8 éléments. Pour conserver le caractère 'list' d'un ensemble de valeurs introduit dans un autre, il sera nécessaire de faire appel à une notion avancée (que l'on verra ultérieurement) : les références.

des parenthèses vides pour définir une liste vide :

- **() définit une liste ne contenant aucun élément.**

```
#!/usr/bin/perl
use warnings;
use strict;

my @liste_vide = ();
```

Comment conserver des listes ?

Auteurs : 2Eurocents ,

Les listes que l'on vient de voir sont un peu volatiles... nous ne les avons pas encore rangées dans des variables. Elles ne peuvent pas être conservées dans une variable scalaire ordinaire. Perl a besoin de types de variables spéciaux pour gérer les listes. Pour les listes simples, le type utilisé est le type tableau, ou vecteur.

Comment définir et utiliser un tableau ?

Auteurs : 2Eurocents ,

Un tableau est une variable dont le sigil (le caractère qui précède le nom) est @ :

```
my @tableau = (1, 2, 3);
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

L'usage de ce tableau se fait de deux manières, selon que l'on veut le manipuler dans sa globalité, ou bien accéder à un élément particulier. L'accès global au tableau se fait avec la notation en @ :

```
@tableau = ();              # pour vider un tableau
@tableau = ( 1, 2, 3 );     # pour affecter une liste à un tableau

# pour affecter le contenu d'un tableau à un
# autre tableau (attention, il s'agit d'une duplication du contenu)
@tableau = @list;

# pour affecter le contenu de trois tableaux dans un autre
@tableau = ( @liste1, @liste2, @liste3 );

# pour ajouter des listes ou des tableaux à un
# tableau, avant et après les valeurs actuelles
@tableau = ( @liste_avant, @tableau, @liste_apres );
```

Une erreur fréquente chez les débutants consiste à affecter un tableau à une valeur scalaire :

```
$i = @tableau;
```

Ce qui ne fonctionne pas comme on pourrait le souhaiter... En contexte scalaire (quand on l'affecte à un scalaire, ou bien lorsque l'on l'écrit scalar @tableau), un tableau retourne le nombre d'éléments qu'il contient. En contexte scalaire de chaîne, le tableau retourne la liste de ses éléments, séparés par des blancs. Comment forcer ce contexte de scalaire de chaîne ? Tout simplement en encadrant le tableau par des doubles quotes :

```
$contenu = "@tableau";
```

Mais comment accéder à un élément en particulier ?

Auteurs : 2Eurocents ,

Un tableau contient des scalaires. Pour en récupérer un en particulier, il faut indiquer que l'on veut un scalaire, donc utiliser le sigil $ devant le nom du tableau et indiquer entre crochets lequel en particulier (rang de rangement dans le tableau, le premier ayant pour rang 0).
$tableau[5] correspond donc au scalaire se trouvant en 6e position du tableau. Les indices de tableau sont donc des entiers positifs, commençant à 0.
Il est possible d'utiliser des entiers négatifs pour effectuer un parcours à rebours du tableau - donc en commençant par la fin :

- **$tableau[-1] désigne ainsi le dernier élément du tableau ;**
- **$tableau[-2] l'avant-dernier ;**
- **$tableau[-3] l'antépénultième, etc.**

Comment accéder à tous les éléments ?

Auteurs : Djibril ,

Pour obtenir le contenu d'une liste, il existe trois fonctions nous facilitant la vie : foreach, for et each.

- **foreach : cette instruction permet de parcourir une liste. Son implémentation est optimisée dans l'interpréteur Perl et est plus efficace qu'un for ;**
- **for : cette instruction permet de parcourir une liste à partir d'indices ;**

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

- **each: utilisable avec les listes depuis la version perl 5.12, elle permet de parcourir une liste et retourne l'indice de l'élément et sa valeur (et non la clef et sa valeur comme c'est le cas dans les listes associatives).**

Voici des exemples simples.

```
#!/usr/bin/perl
use warnings;
use strict;

my @fruits = qw/ Orange Mangue Fraise Pomme Kiwi Ananas Prune/;

print "Utilisation des foreach\n";
foreach my $fruits (@fruits) {
  print "$fruits\n";
}

print "\nUtilisation de for\n";
for ( my $i = 0; $i < scalar(@fruits); $i++ ) {
  print "$fruits[$i]\n";
}

print "\nUtilisation de each (unique depuis perl 5.12)\n";
while ( my ( $indice, $valeur ) = each @fruits ) {
  print "$indice : $valeur\n";
}
```

Résultat

```
Utilisation de foreach
Orange
Mangue
Fraise
Pomme
Kiwi
Ananas
Prune

Utilisation de for
Orange
Mangue
Fraise
Pomme
Kiwi
Ananas
Prune

Utilisation de each (uniquement depuis perl 5.12)
0 : Orange
1 : Mangue
2 : Fraise
3 : Pomme
4 : Kiwi
5 : Ananas
6 : Prune
```

Comment gérer le nombre d'éléments du tableau ?

Auteurs : 2Eurocents ,

C'est tout simple, on ne le gère pas, Perl s'occupe de tout !
Pour connaître le nombre d'éléments on utilise la fonction scalar.

```
my $nbr = scalar @tableau;
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Il est aussi possible de connaître l'indice du dernier élément du tableau, grâce à une combinaison spéciale de sigils devant le nom de tableau.

```
my @tableau = ( 1, 56, 'ClaudeLELOUP' );
my $indice_de_fin = $#tableau;
print $indice_de_fin; # 2
```

$#tableau retourne donc le rang du dernier élément de tableau (pour mémoire, c'est un nombre, donc un scalaire, d'où le $, et c'est un indice, donc un numéro, d'où le # qui signifie numéro en notation commerciale américaine).
Maintenant, accéder à des éléments hors des bornes du tableau ne pose pas de problème non plus. Perl crée automatiquement tous les éléments intermédiaires en leur affectant une valeur indéfinie (undef) :

```
my @liste = ( 1 .. 5 );
$liste[10] = 1;
```

crée une liste de 11 éléments *(1, 2, 3, 4, 5, undef, undef, undef, undef, undef, 1)*.
Pour savoir si un élément de tableau existe réellement, c'est-à-dire s'il a été créé explicitement, et pas par extension de tableau, Perl met à notre disposition la fonction exists() qui retourne vrai si l'élément a bien été créé, faux s'il a été obtenu par extension.
Si vous vous souvenez bien des initialisations de variables, il est possible de laisser une variable non initialisée, voire de la vider en lui affectant la valeur undef. Dans ce dernier cas, s'il s'agit d'un élément de tableau, le test avec la fonction defined() retourne faux, puisque la variable est indéfinie, mais le test avec exists() retourne vrai puisque la variable a été explicitement indéfinie. Il faut donc impérativement faire la distinction entre test de définition (defined()) et test d'existence (exists()).

Jongler avec listes et tableaux : c'est le cirque !

Auteurs : 2Eurocents ,

L'équivalence liste/tableaux, la gestion automatique de la taille des tableaux, l'aplatissement des listes les unes dans les autres permettent à Perl des constructions très intéressantes :

```
($v1, $v2, $v3) =
 (10, 20, 30); # est une affectation de trois valeurs, simultanément. Notez que si vous avez plus d'éléments
# dans la liste de gauche de l'affectation que dans celle de droite, les éléments en trop à gauche recevront la
# Inversement, si vous avez plus d'éléments à droite du signe égal qu'à gauche, les dernières valeurs à droite s

($v1, $v2, $v3) = ($v2, $v3, $v1); # est une façon intéressante de mélanger/
permuter des valeurs de variables scalaires, que vous en
# ayez 2, 3, ou n à échanger (enfin, pas trop, quand même, pour que ça reste lisible et maintenable).

($valeur, @tableau) = @tableau; # est une très jolie façon de supprimer la première valeur d'un tableau, tout en

(@tableau, $valeur) = @tableau; # ne sert à rien, car le tableau en début de liste récupére toutes les valeurs d
# Il ne reste rien pour remplir $valeur, qui vaudra alors undef.

($v1, $v2) = @tableau; # met dans $v1 et $v2 les deux premiers scalaires contenus dans le tableau (sans modifier
ci),
# ce qui est à rapprocher du premier exemple donné ici.

($v1, undef, $v2, undef, undef, $v3) = @tableau; # met dans $v1 la première valeur du tableau, dans $v2 la trois
# Les autres scalaires du tableau sont ignorés.

my ($a, $b, $c); # et
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
my ($a, $b, $c) =
 (1, 2, 3); # sont les seules façons correctes de faire des déclarations multiples et simultanées de variables.
```

Les listes, c'est bon, reprenez-en une tranche !

Auteurs : 2Eurocents ,

Si vous souhaitez manipuler des portions de listes, il existe un mécanisme délicatement nommé "tranches" qui vous permet de récupérer une liste qui est un sous-ensemble d'une autre. Ce mécanisme a une syntaxe un peu spéciale mais tout à fait explicite. Si l'on travaille sur une liste (ensemble exprimé entre parenthèses), il suffit de suffixer la liste par une expression des indices des valeurs souhaitées, entre crochets.
Cette expression peut être constituée d'indices distincts, séparés par des virgules, et/ou indiquer des intervalles d'indices au moyen de l'opérateur d'intervalles .. :

```
@l = ('a', 'b', 'c', 'd', 'e', 'f')[2,4,0]; # remplit @l avec la liste ('c', 'e', 'a')
@l = ('a', 'b', 'c', 'd', 'e', 'f')[0, 2..4]; # remplit @l avec la liste ('a', 'c', 'd', 'e')
```

Si l'on travaille sur un tableau, dont on souhaite récupérer une tranche, la spécification d'indices a la même forme, mais le nom de tableau doit être précédé du sigil @, car on n'accède pas à un scalaire, mais à un tableau, constitué par cette fameuse tranche :

```
@l = ('a', 'b', 'c', 'd', 'e', 'f');
@sl = @l[2,4,0];
```

Il est même possible de faire des remplacements dans un tableau, au moyen du mécanisme de tranches :

```
@l = ('a', 'b', 'c', 'd', 'e', 'f');
@l[2,4,0]=(1,2,3);
```

modifie la liste @l qui vaut alors (3, 'b', 1, 'd', 2, 'f').

```
@l = qw(a b c d e f);
@l[1..3]=(1,2,3);
```

modifie la liste @l qui vaut alors ('a', 1, 2, 3, 'e', 'f'). Pire encore, les mécanismes de tranches et d'aplatissement des listes permettent d'agrandir ou de rétrécir un tableau par le milieu :

```
@l = qw(a b c d e f);

# modifie la liste @l pour qu'elle vaille ('a', 'b', 'c', 1, 2, 3, 'e', 'f')
# en gros, on remplace l'élément d'indice 3 par la liste (1, 2, 3).
@l = ( @l[ 0 .. 2 ], ( 1, 2, 3 ), @l[ 4 .. 5 ] );

@l = qw(a b c d e f);
@l = ( @l[ 0 .. 1 ], @l[ 4 .. 5 ] );    # supprime 'c' et 'd' dans le tableau @l..
```

Et les fonctions sur les listes et les tableaux ?

Auteurs : 2Eurocents , Djibril ,

Perl contient de nombreuses fonctions gérant les listes et les tableaux. En voici quelques-unes des plus utiles :

- **qw**

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

permet de créer une liste à partir d'une chaîne de caractères. La chaîne est découpée selon les blancs, les tabulations et les sauts de ligne. Chaque "mot" devient alors un élément de liste.
Attention : seuls les espaces, tabulations, et sauts de ligne sont pris en compte. Aucune protection par des quotes ou doubles quotes n'est efficace. On peut ainsi écrire :

```
@l = qw /Cela est beau et bon/;
```

plutôt que

```
@l = ('Cela', 'est', 'beau', 'et', 'bon');
```

- **shift**

permet de supprimer le premier élément d'une liste et de retourner sa valeur. Ainsi, $val = shift @t; est finalement équivalent à ($val, @t) = @t;

- **unshift**

permet d'ajouter un ou plusieurs éléments au début d'un tableau. unshift (@t, 1, 2, 3); ajoutera (1, 2, 3) au début du tableau @t.

- **pop**

permet de supprimer le dernier élément d'une liste et de retourner cette valeur.

```
my @mois = qw/ mars avril septembre decembre/;
my $dernier = pop @mois;
# $dernier =  decembre
# @mois    =  mars avril septembre
```

- **unpop**

à l'inverse, permet d'ajouter un ou plusieurs éléments en fin de tableau.

- **delete**

permet de supprimer un élément d'un tableau. Mais attention, la valeur supprimée est positionnée à undef, il rend donc vrai au test not defined (), et un comptage du nombre d'éléments dans le tableau avec scalar comptabilisera également les éléments supprimés.

```
#!/usr/bin/perl
use strict;
use warnings;

my @mois = qw/ mars avril septembre décembre/;

print 'Il y a ', scalar @mois, " élément dans le tableau\n";
delete $mois[1];  # Supprimera avril, MAIS, le remplacera par undef
print 'Il y a ', scalar @mois, " élément dans le tableau\n";
print "\n\n\n";

# Le tableau ressemble à qw/ mars undef septembre décembre /
foreach my $elt ( @mois ) {
  print "- $elt\n";
}
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
Il y a 4 élément dans le tableau
Il y a 4 élément dans le tableau

- mars
Use of uninitialized value $elt in concatenation (.) or string at xxxx\test.pl line 14.
-
- septembre
- décembre
```

Vous remarquerez que le nombre d'éléments du tableau n'a pas changé, qu'avril a bien été remplacé par undef. De plus, le message d'avertissement *"Use of uninitialized value $elt in concatenation (.) or string at"* signifie que Perl a lu la case où il y avait avril, mais comme elle est à undef, il s'agit d'une valeur non initialisée. Pour pallier cet inconvénient il aurait fallu faire un test avec defined.
Soyez donc prudent ! Regardez la fonction splice qui peut être plus adaptée à vos besoins.

- **split**

permet de découper une chaîne selon des marqueurs spécifiques et retourne la liste des éléments résultant de ce découpage. Le marqueur spécifique est indiqué sous la forme d'une expression rationnelle. Sans entrer dans le détail de ceeux-ci, voici des exemples :

```
split / \t\n/, "chaîne à découper";
  # effectue le même travail que qw sur la "chaîne à découper".
split /:/, "root:password:0:0:/bin/bash"; # effectue le découpage d'une ligne de fichier de mots de passe *n*x s
```

- **join**

effectue la tache inverse de split. Elle rassemble les éléments de la liste fournie en les concaténant, tout en les séparant par la chaîne indiquée comme premier paramètre. Ainsi join ('...', 1, 2, 3); créera la chaîne "1...2...3".

- **splice (tableau, début, nombre)**

Supprime x éléments d'un tableau à partir de l'indice début spécifié.

```
my @mois = qw/ mars avril septembre décembre/;
print 'Il y a ', scalar @mois, " élément dans le tableau\n";
splice @mois,1,2;  # Supprimera définitivement avril et septembre
print 'Il y a ', scalar @mois, " élément dans le tableau\n";
print "\n";

# Le tableau ressemble à qw/ mars undef septembre décembre /
foreach my $elt ( @mois ) {
  print "- $elt\n";
}
```

```
Il y a 4 élément dans le tableau
Il y a 2 élément dans le tableau

- mars
- décembre
```

Quelques fonctions surpuissantes !

Auteurs : 2Eurocents , Djibril ,

- **sort**

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

C'est une fonction interne de Perl permettant de trier une liste. Elle retourne une liste triée.

```
my @mois       = qw/ mars avril septembre decembre/;
my @mois_tries = sort @mois;                          # avril decembre mars septembre
```

Les deux listes peuvent être distinctes. La liste d'origine peut être écrasée par le résultat du tri si elle est destinataire de l'affectation.
La fonction sort peut prendre un argument spécial qui est le bloc de comparaison à effectuer entre les éléments de la liste à trier. Il est ainsi possible de trier selon des critères tout à fait personnalisés. Par défaut, le tri se fait dans l'ordre "lexicographique" (ordre alphabétique, étendu aux nombres dont les chiffres sont traités comme des caractères et non comme des nombres). Le bloc de comparaison personnalisé doit être précisé entre accolades, avant la liste d'éléments à trier, sans être séparé de celle-ci par une virgule.
Pour la comparaison, ce bloc utilise deux variables internes de Perl, $a et $b, qui ne sont définies que dans ce bloc et masquent toute variable $a ou $b propre à l'utilisateur. Ce bloc effectue donc un test quelconque, basé sur $a et $b, dont le résultat peut prendre trois valeurs :

- **positif si $a est avant $b dans l'ordre de tri souhaité ;**
- **nul si $a et $b sont équivalents ;**
- **négatif si $a est après $b dans l'ordre souhaité.**

Ces trois valeurs de résultat de test correspondent aux résultats des opérateurs de test <=> et cmp. Ainsi, pour un tri lexicographique du tableau @t, on peut faire :

```
my @mois_tries = sort @mois;
```

ou bien

```
my @mois_tries = sort { $a cmp $b } @mois;
```

Pour un tri lexicographique inversé, on peut aussi bien écrire :

```
my @mois_tries = reverse sort @mois;
```

que

```
my @mois_tries = sort { $b cmp $a } @mois;
```

bien que cette méthode soit moins performante que la précédente.
Pour un tri numérique :

```
@l = sort { $a <=> $b } @t;
```

Pour un tri un peu spécial, en supposant que @t contienne des indices d'un tableau et que l'on souhaite un tri de ces indices selon les valeurs numériques du tableau :

```
@l = sort { $tableau[$a] <=> $tableau[$b] } @t;
```

@l contient alors les valeurs de @t triées dans un ordre tel, qu'elles indicent des valeurs croissantes dans @tableau. C'est très indirect et ce n'est pas forcément très naturel au début, mais c'est extrêmement puissant. Et pour finir, bien que l'on n'ait pas encore abordé la définition de fonctions personnalisées, il est possible d'appeler une fonction définie par le programmeur :

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
@l = sort { mafonction($a, $b) } @t;
```

- **map**

C'est une fonction interne de Perl permettant de parcourir une liste et d'appliquer un traitement à chaque élément de la liste.
Elle retourne la liste des éléments traités.
Comme pour la fonction sort, il faut fournir un bloc de traitement, exprimé entre accolades et qui ne sera pas séparé de la liste à traiter. Le traitement effectué peut être totalement quelconque, porter sur des chaînes, des nombres ou des listes, utiliser une fonction interne à Perl ou une fonction utilisateur, etc. Dans ce bloc, l'élément en cours de traitement se trouve dans la variable interne à Perl nommée $_.

```
@l = map { $_ + 1 } @t;
```

et @l contiendra tous les éléments de @t augmentés de 1.

```
@l = map { ($_) x 3 } @t;
```

et @l contiendra tous les éléments de @t, mais dupliqués trois fois.

```
$i=0;
@l=map { $i += $_ } @t;
```

et @l contiendra la somme des éléments de @t d'indice inférieur ou égal.
Si le bloc de traitement effectue des modifications sur $_, celles-ci seront répercutées sur la liste d'origine. La plus grande rigueur est donc de mise...

```
@l = map { $_ .= '+' } @t;
```

suffixe toutes les valeurs de @t avec un "+". Les nouvelles valeurs se trouvent aussi bien dans @t que dans @l.

```
my @mois = qw/ mars avril septembre decembre/;
my @mois_tries = map { suffix($_); } @mois;

# @mois_tries contiendra =mars= =avril= =septembre= =decembre=
sub suffix {
  my $argument = shift;
  $argument = '=' . $argument . '=';
  return $argument;
}
```

A chaque élément de notre liste, un traitement sera effectué. La valeur sera encadrée du signe "=".

- **grep**

C'est une fonction interne de Perl permettant de parcourir une liste et de sélectionner les éléments en fonction de critères définis. Pour les Linuxiens, c'est le principe grep !!!
La fonction parcourt l'intégralité de la liste fournie en paramètres et lui applique un traitement de sélection fourni par l'utilisateur. Elle retourne la liste des éléments qui répondent aux critères. Comme pour les fonctions sort et map, il faut fournir un bloc de traitement, exprimé entre accolades et qui ne sera pas séparé de la liste à traiter. Dans le bloc de sélection, l'élément en cours de traitement se trouve dans la variable interne à Perl nommée $_.

```
@l = grep { $_ != 0 } @t;
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

et @l contiendra tous les éléments non nuls de @t.

```
@l = grep { /mot-clef/ } @t;
```

et @l contiendra tous les éléments de @t dans lesquels "mot-clef" est présent, grâce au mécanisme des expressions rationnelles. Voici, à titre d'exemple un traitement plus complexe (bien que tout à fait artificiel, Perl ayant des mécanismes que nous allons voir bientôt qui permettent de le faire encore plus efficacement) :

```
# Préparation des données
my @eleves  = qw /Valentine Chloé Sophie Olivier Jérôme Samuel/;
my @notes   = qw/ 8 16 19 12 6 15/;
my @indices = (0..$#eleves);
# Fin de la préparation...
# Relevé des notes permettant le passage en classe supérieure
my @passage = grep { $notes[$_]>=10 } @indices;
# liste des élèves admis
my @passent = map { $eleves[$_] } @passage;
```

Cependant, ce traitement est vulnérable car le lien entre les notes et les élèves est assez lâche. Il s'agit d'un indice, une position dans le tableau. Si le tableau est trié, tout le traitement s'effondre. C'est pourquoi il existe en Perl une structure de données plus efficace pour associer des valeurs entre elles, les listes associatives.

lien : Tri en Perl d'après les mongueurs

lien : A Fresh Look at Efficient Perl Sorting

lien : Traduction de A Fresh Look at Efficient Perl Sorting

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > S'initier à Perl > Les listes associatives

Qu'est-ce qu'une liste associative ?

Auteurs : 2Eurocents ,

Une liste associative, ou table de hachage est un type de variable particulier contenant des scalaires associés entre eux par paires.
Chaque paire est constituée d'un premier élément, nommé "clef", qui servira de repère pour retrouver la valeur dans la table associative. Le second élément de la paire est la "valeur" qui sera stockée dans la table.
A chaque clef correspond une valeur. Il ne peut pas y avoir deux clefs identiques. Par contre, deux valeurs identiques peuvent être repérées par deux clefs distinctes.
La définition d'une paire ayant une clef déjà utilisée remplace la valeur associée à cette clef par la nouvelle valeur fournie.
Une liste associative peut s'exprimer par une ou plusieurs paires de valeurs, séparées par des virgules, mises entre parenthèses comme :

```
('a', 'chaîne a', 'b', 'chaîne b', 'zz', 'chaîne zz')
```

qui est l'expression d'une liste associative rassemblant les couples :

```
('a', 'chaîne a'), ('b', 'chaîne b'), ('zz', 'chaîne zz')
```

Le problème de cette notation est que l'on ne fait pas forcément la distinction entre listes et listes associatives. Cela peut être confortable pour passer d'une forme à l'autre, mais cela peut aussi être source d'erreur (notamment si l'on n'a pas mis un nombre pair de valeurs dans la table). C'est pourquoi il existe une autre notation qui met en évidence le caractère associatif des listes hachées :

```
('a' => 'chaîne a', 'b' => 'chaîne b', 'zz' => 'chaîne zz')
```

Ainsi, l'association clef/valeur est explicitement mise en évidence. Il est bien sûr nettement préférable d'utiliser cette notation. La nature des scalaires contenus dans une liste associative n'a absolument aucune espèce d'importance pour Perl. De même, la nature des valeurs utilisées comme clefs n'est pas critique. Les clefs doivent simplement correspondre à une chaîne de caractères, au sens large.
Perl est assez permissif sur la syntaxe des clefs et autorise, soit leur écriture entre quotes simples ou doubles, soit leur écriture sans quotes si la chaîne est composé exclusivement des lettres A à Z, a à z, des chiffres de 0 à 9 et du tiret de soulignement "_". De la même manière que les listes simples, les listes associatives s'aplatissent les unes dans les autres :

```
(a => 'chaîne a', b => 'chaîne b', (A => 'chaîne A', B => 'chaîne B', C => 'chaîne C'),
 a => '2eme chaîne a');
```

définit une liste associative de cinq paires, dont les clefs sont 'a', 'b', 'A', 'B' et 'C'. La sixième paire définie utilisant de nouveau la clef 'a', la valeur associée 'chaîne a' sera remplacée par '2eme chaîne a'. La liste associative incluse a été aplatie et ses paires se retrouvent au même niveau que les paires de la liste principale.
Pour inclure une liste associative dans une autre, en lui conservant son caractère "liste", il sera nécessaire de faire appel à une notion avancée (que l'on verra ultérieurement) : les références.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Comme pour les listes simples, une liste associative vide est définie par une paire de parenthèses vides : () définit une liste (simple ou associative) ne contenant aucun élément.

Comment conserver des listes associatives ?

Auteurs : 2Eurocents ,

Les listes associatives que l'on vient de voir sont elles aussi un peu volatiles... nous ne les avons toujours pas rangées dans des variables.
Elles ne peuvent pas être conservées dans une variable scalaire ordinaire et les ranger dans une table ordinaire n'aurait pas de sens car on perdrait le caractère associatif. Perl utilise un type de variable spécial pour gérer les listes associatives.

Comment définir et utiliser une table associative ?

Auteurs : 2Eurocents ,

Une table associative est une variable dont le sigil (le caractère qui précède le nom) est % :

```
my %association = (1 => 'aaa', 2 => 'bbb', 3 => 'ccc');
```

L'usage de cette table se fait de deux manières, selon que l'on veut la manipuler dans sa globalité, ou bien accéder à un élément particulier. L'accès global au tableau associatif se fait avec la notation en %.

```
%hash = (); #pour vider un tableau de hachage.
%hash =
 (1 => 'premier', 2 => 'deuxieme', 3 => 'troisieme'); # pour affecter une nouvelle liste associative.
%hash = %list; # pour affecter le contenu d'un tableau associatif à un autre tableau de hachage (attention, il s

# pour affecter le contenu de trois tableaux associatifs dans un autre. Les clefs présentes dans plusieurs table
# simultanément ne se retrouveront qu'en un exemplaire dans le hachage résultant avec pour valeur associée la de
%hash = (%hash1, %hash2, %hash3);

# pour affecter une liste simple à une liste associative. Les éléments de la liste simple sont pris
# par paires, le premier de chaque paire étant utilisé comme clef, le second comme valeur.
%hash = @list;
```

Par contre, l'évaluation d'une table associative en contexte scalaire n'est, à priori, pas une idée pertinente.

Et comment récupére-t-on le nombre d'éléments ?

Auteurs : 2Eurocents ,

Puisque l'on ne peut pas évaluer la liste associative en contexte scalaire, contrairement aux listes simples, comment peut-on récupérer le nombre d'éléments d'une liste associative ?
Perl fournit deux fonctions particulières d'accès aux listes associatives : keys() et values().

- **keys() retourne une liste simple contenant toutes les clefs de la liste associative passée en paramètres ;**
- **values() retourne une liste simple contenant toutes les valeurs de la liste associative passée en paramètres.**

Pour connaître le nombre d'éléments de la liste associative, il faut trouver le nombre d'éléments d'une de ces deux listes simples. Par contre, il est important de savoir que l'ordre des éléments dans les listes retournées par keys() et values() n'a rien à voir avec l'ordre dans lequel les éléments ont été ajoutés au hachage.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Il s'agit d'un ordre qui permet à Perl d'optimiser ses recherches lorsque l'on souhaite accéder à un élément en particulier.

Comment accéder à un élément en particulier ?

Auteurs : 2Eurocents ,

Un tableau associatif contient des scalaires. Pour en récupérer un en particulier, il faut indiquer que l'on veut un scalaire, donc utiliser le sigil(1) $ devant le nom du tableau de hachage, et indiquer entre accolades la clef de celui que l'on souhaite récupérer.
$hash{aaa} correspond donc au scalaire se trouvant associé à la clef "aaa".
Notez bien que si la clef est bien une chaîne de caractères, il n'est pas nécessaire de la mettre entre quotes tant qu'elle ne contient pas d'espace ni de caractère susceptible de gêner l'interpréteur.
Si la clef fournie n'existe pas, Perl créera la paire d'éléments avec la clef d'une part, et la valeur "undef" d'autre part. Il est donc prudent, plutôt que de tenter d'accéder directement à des éléments dont on ne sait pas s'ils existent ou non, de les tester au préalable grâce à la fonction exists(). Ce qui ne vous empêche pas par ailleurs, de faire des tests sur la définition valide de son contenu par la fonction defined().

```
my %identite = (
  nom    => 'dupond',
  prenom => 'jean',
);

if ( exists $identite{nom} ) {
  print "Nom : $identite{nom}\n";
}

if ( exists $identite{age} ) {
  print "Age : $identite{age}\n";
}
```

```
Nom : dupond
```

- **Un sigil est le premier caractère d'un identificateur en Perl. Il est non alphanumérique et dénote son type.**

Comment accéder à tous les éléments ?

Auteurs : Djibril ,

Pour lister le contenu d'un hash, il existe trois opérateurs nous facilitant la vie : keys, values et each.

- **keys retourne une liste simple contenant toutes les clefs de la liste associative ;**
- **values retourne une liste simple contenant toutes les valeurs de la liste associative ;**
- **each permet d'effectuer une itération sur l'ensemble des éléments d'un hash sans avoir à procéder à un accès préalable aux clefs.**

Voici des exemples simples.

```
#!/usr/bin/perl
use warnings;
use strict;

my %hash = (
  'Nom'        => 'Dupond',
  'Prenom'     => 'Jean',
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
  'Adresse'    => '10 rue de Rome',
  'Ville'      => 'Paris',
  'CodePostal' => '75000'
);

print "Utilisation des clefs\n";
foreach my $clef ( keys %hash ) {
  print "$clef : $hash{$clef}\n";
}

print "\nUtilisation de each\n";
while ( my ( $clef, $valeur ) = each %hash ) {
  print "$clef : $valeur\n";
}

print "\nAffichage uniquement des valeurs\n";
foreach my $valeur ( values %hash ) {
  print "$valeur\n";
}
```

```
Utilisation des clefs
Ville : Paris
Adresse : 10 rue de Rome
CodePostal : 75000
Prenom : Jean
Nom : Dupond

Utilisation de each
Ville : Paris
Adresse : 10 rue de Rome
CodePostal : 75000
Prenom : Jean
Nom : Dupond

Affichage uniquement des valeurs
Paris
10 rue de Rome
75000
Jean
Dupond
```

Il est impossible de trouver une clef à partir d'une valeur.

Comment supprimer un élément de la liste temporairement ou non ?

Auteurs : Djibril ,

Il est possible de supprimer définitivement un élément de notre liste associative avec la fonction delete.
Depuis Perl 5.12, nous avons la possibilité de supprimer un élément de notre liste associative localement dans notre programme. Pour ce faire, il faut associer à delete la fonction local : delete locale $hash{clef}; Voici un exemple :

```
#!/usr/bin/perl
use warnings;
use strict;

my %hash = (
  'Nom'        => 'Dupond',
  'Prenom'     => 'Jean',
  'Adresse'    => '10 rue de Rome',
  'Ville'      => 'Paris',
  'CodePostal' => '75000'
);
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
print "\nListons le contenu du hash\n";
while ( my ( $clef, $valeur ) = each %hash ) {
  print "$clef : $valeur\n";
}

# supprimons localement 2 éléments
{
  delete local $hash{Nom};
  delete local $hash{CodePostal};

  print "\nListons localement le contenu du hash\n";
  while ( my ( $clef, $valeur ) = each %hash ) {
    print "$clef : $valeur\n";
  }

}

print "\nListons le contenu du hash - version 2\n";
while ( my ( $clef, $valeur ) = each %hash ) {
  print "$clef : $valeur\n";
}
```

Résultat

```
Listons le contenu du hash
Ville : Paris
Adresse : 10 rue de Rome
CodePostal : 75000
Prenom : Jean
Nom : Dupond

Listons localement le contenu du hash
Ville : Paris
Adresse : 10 rue de Rome
Prenom : Jean

Listons le contenu du hash - version 2
Ville : Paris
Adresse : 10 rue de Rome
CodePostal : 75000
Prenom : Jean
Nom : Dupond
```

Encore une tranche ?

Auteurs : Djibril ,

Une tranche est une façon commune d'accéder à une liste ou liste associative afin d'en prendre plusieurs éléments simultanément. Les tranches permettent de compacter notre code en évitant des appels individuels aux éléments.

```
#!/usr/bin/perl
use strict;
use warnings;

my @liste = ( 1 .. 10 );
my @tranche_liste = @liste[ 2, 4, 0 ];
print "Tranche de liste : @tranche_liste\n";

my %liste_associative = (
  nom     => 'dupond',
  prenom  => 'jean',
  age     => '22',
  sexe    => 'masculin',
  langage => 'perl',
);
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
my @tranche_liste_associative = @liste_associative{ 'prenom', 'age', 'langage' };
print "Tranche de liste associative : @tranche_liste_associative\n";
```

Résultat des tranches

```
Tranche de liste : 3 5 1
Tranche de liste associative : jean 22 perl
```

Si vous êtes troublé par l'usage d'un "@" ici sur une tranche de hachage au lieu d'un "%", pensez-y ainsi. Le type de parenthésage (avec des crochets ou des accolades) décide si c'est un tableau ou un hachage qui est examiné. D'un autre côté, le symbole en préfixe ("$" ou "@") du tableau ou du hachage indique si vous récupérez une valeur simple (un scalaire) ou une valeur multiple (une liste).

Des fonctions, pour les listes et les tableaux associatifs ?

Auteurs : 2Eurocents ,

Les fonctions shift(), unshift(), pop() et unpop() n'ont pas de raison d'être utilisées avec les tables associatives, l'ordre des éléments étant inconnu et déterminé automatiquement par Perl. Les fonctions split() et join() aussi n'ont plus de sens... mais que reste-t-il alors ?
delete() supprime toujours l'élément fourni.

```
delete $h{clef}
```

Il supprime l'élément de la table de hachage %h, dont la clef est 'clef'. Sa valeur n'est pas seulement mise à undef. Sa clef est également supprimée et les tests avec exists $h{clef} seront faux.
reverse() permet d'inverser une liste. Dans le cas d'une table de hachage, il fait un échange entre les clefs et les valeurs. Les clefs deviennent les valeurs et les valeurs deviennent les clefs. Cela fonctionne très bien si les valeurs sont uniques. Dans le cas contraire, on ne peut pas prévoir quels seront les couples clef/valeur qui seront conservés pour les valeurs de départ en double.
sort() ne peut pas trier une liste associative, mais permet, grâce à son bloc d'instructions, toutes sortes de tris sur les clefs ou sur les valeurs. Ainsi, pour un tri lexicographique des clefs de %t, on peut faire :

```
@l = sort { $a cmp $b } keys %t;
```

qui fournit une liste des clefs triées. Pour un tri lexicographique des valeurs de %t, on peut faire :

```
@l = sort { $a cmp $b } values %t;
```

qui fournit une liste des valeurs triées - mais l'on a perdu les clefs associées. Pour un tri lexicographique des clefs de %t, selon leurs valeurs associées, on peut faire :

```
@l = sort { $t{$a} cmp $t{$b} } keys %t;
```

qui fournit une liste des clefs triées selon les valeurs associées croissantes - on a bien trié une liste simple de clef, mais le bloc d'instructions faisait la comparaison sur les valeurs liées. map() et grep() ne parcourront pas directement une table associative mais permettent tout traitement sur les listes de clefs ou les listes de valeurs. Pour reprendre l'exemple scolaire qui clôturait le chapitre sur les listes, en voici une réécriture plus pertinente au moyen de listes associatives :

```
# Préparation des données
my %moyennes = (
 'Valentine' => 8,
 'Chloé' => 16,
 'Sophie' => 19,
 'Olivier' => 12,
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
  'Jérôme' => 6,
  'Samuel' => 15
);

# Fin de la préparation...
# Relevé des notes permettant le passage en classe supérieure
# et liste des élèves admis, traitement simultané
my @passent = grep { $moyenne{$_} >= 10 } keys %moyennes;
```

Cette réécriture du traitement est plus efficace, plus facile à maintenir et plus sûre que celle basée sur les tableaux. En outre, nous commençons à peine à ébaucher la puissance de Perl. Nous avons réalisé ce type de traitement de sélection, simplement en manipulant des données, sans construction de traitements complexes. Nous allons maintenant pouvoir ajouter tout ce qui est nécessaire au contrôle fin de l'exécution des traitements...

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > S'initier à Perl > Structures de contrôle

Tout en bloc !

Auteurs : 2Eurocents ,

En Perl, il est possible et même souhaitable d'organiser le code en blocs d'instructions. Ces blocs seront exécutés ou non en fonction de différentes conditions (que l'on nomme alternatives) ou pourront être répétés autant que de besoin (itératives).
Un bloc débute par une accolade ouvrante et se termine par une accolade fermante.
Les blocs peuvent être imbriqués les uns dans les autres. Un bloc peut contenir indifféremment d'autres blocs ou des commandes isolées. Il est de coutume d'indenter les blocs, c'est-à-dire d'introduire un ou plusieurs espaces de marge à gauche des instructions contenues dans les blocs. Plus il y a des niveaux d'imbrication, plus on introduit des espaces. Ceci permet une lecture plus aisée de la structure du code et permet même à certains éditeurs de texte de la mettre en valeur par des barres verticales reliant le début à la fin du bloc, ou par l'apparition de boutons permettant de masquer/ démasquer le bloc à volonté (cas de Scintilla Text Editor, logiciel libre d'édition de code source).

Visibilité réduite à 6 miles...

Auteurs : 2Eurocents ,

La notion de bloc d'instructions limite la visibilité des variables. En effet, quel que soit leur type (scalaire, tableau, table associative), les variables ne sont visibles - accessibles - qu'à l'intérieur du bloc où elles sont définies. Elles sont donc utilisables à l'intérieur de la paire d'accolades contenant leur déclaration ou à l'intérieur des blocs que celle-ci contient. En outre, toute redéclaration d'une variable de même nom à l'intérieur d'un bloc masquera automatiquement la valeur que la variable avait acquise à l'extérieur de ce bloc. La variable reprendra sa valeur d'origine aussitôt que l'exécution du bloc contenant la redéclaration sera terminée. On parle alors de variables locales, la portée de la variable (là où l'on peut l'utiliser) étant limitée à ce seul bloc d'instructions.

Longue portée...

Auteurs : 2Eurocents ,

La portée et la visibilité que l'on vient de définir sont en fait valables pour les variables déclarées au moyen du mot-clef my. Celui-ci limite la portée de la variable au bloc en cours d'exécution. Pour qu'une variable soit accessible hors de ce bloc, il existe deux solutions :

- **définir cette variable hors de tout bloc d'instructions. Elle est alors disponible dans tout le script ;**
- **déclarer cette variable avec le mot-clef our. Elle est alors disponible dans tout le script et peut même - sous certaines conditions (cas des modules) - être accédée depuis l'extérieur.**

Quelle alternative ?

Auteurs : 2Eurocents ,

Il y a souvent besoin de réaliser un choix entre deux blocs à traiter selon la valeur d'une condition. C'est ce que l'on appelle une alternative.
En Perl, une condition est une expression, constituée de variables, de valeurs, d'opérateurs de test (<, >, ne, !=, le, ge, cmp...) et éventuellement d'opérateurs de combinaison logique (not, and, or, xor). Le tout doit être habilement mis entre parenthèses afin de garantir les associations de tests souhaitées.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Perl met à notre disposition deux types d'alternative :
- **l'instruction if, qui représente l'alternative favorable - quand une condition est remplie ;**
- **l'instruction unless, qui représente l'alternative défavorable - quand une condition n'est pas remplie.**

La syntaxe est simple :

```
if ( condition ) {
  # bloc d'instructions à exécuter si la condition est vraie
}
```

Pour le cas où l'on souhaite une alternative complète, on peut même faire :

```
if ( condition ) {
  # bloc d'instructions à exécuter si la condition est vraie
} else {
  # bloc d'instructions à exécuter si la condition est fausse
}
```

Et pour les cas de suites d'alternatives imbriquées :

```
if ( condition_primaire ) {
  # bloc d'instructions à exécuter si la condition primaire est vraie
}
elsif (nouvelle_condition) {
  # bloc d'instructions à exécuter si la nouvelle condition est vraie
  #                               alors que la condition primaire est fausse
}
else {
  # bloc d'instructions à exécuter si la nouvelle condition est fausse
  #                               alors que la condition primaire est fausse
}
```

Il est ainsi possible de cascader les elsif dans chaque condition fausse. L'instruction unless fonctionne exactement comme le if, simplement sa condition est inversée. Ainsi, il est équivalent d'écrire :

```
unless ( condition ) { ... }
# et
if ( ! ( condition ) ) { ... }
```

Condition ternaire, opérateur ? :

Auteurs : Djibril ,

Il est possible en Perl d'utiliser les conditions ternaires si vous souhaitez faire des if/elsif/elsif/else. Son utilisation est pratique du fait qu'on ait moins de lignes de code à écrire, que la visibilité du code peut être meilleure et cela nous oblige à faire un else sous peine de recevoir un message d'erreur.
L'écriture est de la sorte : Condition ? exécution si vrai : exécution si faux;
Rien de mieux qu'un exemple.

if/elsif/else
```
#!/usr/bin/perl
use strict;
use warnings;

my $nombre = 20;
my $limite = 20;
my $reponse;
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

if/elsif/else

```
# if/elsif/else
if ( $nombre < $limite ) {
  $reponse = "$nombre plus petit que $limite";
}
elsif  ( $nombre > $limite ) {
  $reponse = "$nombre plus grand que $limite";
}
else {
  $reponse = "$nombre est identique à $limite";
}
print "if/else/else : $reponse\n";
```

Nous allons maintenant faire l'équivalent en utilisant les conditions ternaires.

condition ternaire

```
#!/usr/bin/perl
use strict;
use warnings;

my $nombre = 20;
my $limite = 20;
my $reponse;

               # condition            # reponse
my $reponse2 = $nombre < $limite ? "$nombre plus petit que $limite"
             :  $nombre > $limite ? "$nombre plus grand que $limite"
             :                      "$nombre est identique à $limite";
print "condition ternaire : $reponse2\n";
```

Voilà, c'est simple, clair et en plus, cela nous oblige à utiliser l'équivalent de la condition else. Si vous faites juste :

```
my $reponse2 = $nombre < $limite ? "$nombre plus petit que $limite";
```

vous aurez un message d'erreur Perl de type *syntax error at ....*

Liberté conditionnelle...

Auteurs : 2Eurocents , Djibril ,

Les alternatives que nous venons de voir sont très utiles mais nécessitent l'ouverture et la fermeture de blocs de codes. Cela peut être fastidieux, juste pour une instruction dont l'exécution est conditionnelle. Perl a donc introduit une expression simplifiée pour l'exécution conditionnelle d'une seule instruction :

```
instruction if (condition);
```

ou

```
instruction unless (condition);
```

exécuteront l'instruction indiquée si la condition est vraie (cas du if) ou fausse (cas du unless).
N.B. Bien que Perl donne la possibilité d'écrire votre code comme bon vous semble, il y a quand même des recommandations utiles à respecter.
Ce type d'écriture est recommandé lorsque vous faites appel aux fonctions internes de Perl last, next et redo. Dans les autres cas, il est demandé d'utiliser l'écriture classique. C'est une question de lisibilité améliorée permettant une meilleure maintenance de votre programme.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
foreach my $case ( @tableau ) {
 next if ( condition1 );

 if ( condition2 ) {
  # action 1
  # action 2
 }
}
```

La raideur indigeste... if/elsif/elsif... Switch

Auteurs : 2Eurocents , Djibril ,

Lorsque l'on a une même variable à tester dans différentes conditions, pour vérifier les différentes valeurs possibles, il est nécessaire d'utiliser une batterie de if/elsif/else, ce qui peut paraître rapidement fastidieux et peu clair :

```
if ( $choix == 0 ) {

  # traitement du 0
}
elsif ( $choix == 1 ) {

  # traitement du 1
}
elsif ( $choix == 2 ) {

  # traitement du 2
}
elsif ( $choix == 3 ) {

  # traitement du 3
}
else {

  # traitement par défaut
}
```

- **Utilisateurs d'une version de Perl inférieure ou égale à la 5.8**

Le mécanisme de la sélection (le switch des programmeurs C, la case des pascalistes et des basicois) n'est malheureusement pas disponible pour ces versions de Perl. Toutefois, devant l'intérêt qu'une telle structure présente, une construction non standard a été ébauchée, et elle sera probablement standardisée pour la version 6 du langage.
En attendant, cette fonctionnalité dépend d'un module (une unité d'extension du langage) qui est présent la plupart du temps lors de l'installation de Perl. Il faut l'utiliser avec précaution car elle n'est pas totalement finalisée et possède encore quelques effets de bord, notamment dans les scripts les plus longs. Cependant, voici une réécriture de la sélection, sans les if :

```
use Switch 'Perl6';

# pour avoir le mécanisme de la sélection conforme à ce que permettra Perl6
given ($choix) {
  when 0 {

    # traitement du 0
  }
  when 1 {

    # traitement du 1
  }
  when 2 {
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
    # traitement du 2
  }
  when 3 {

    # traitement du 3
  }
  when /.*/ {

    # traitement par défaut
  }
}
```

C'est un mécanisme à suivre car il réservera de nombreuses (bonnes) surprises.

- **Utilisateurs d'une version de Perl supérieure ou égale à la 5.10**

Depuis la version 5.10, vous pouvez utiliser given/when. Exemple de code :

```
use feature qw/switch/;

given ($string) {
  when (/^abc/) { $abc     = 1; }
  when (/^def/) { $def     = 1; }
  when (/^xyz/) { $xyz     = 1; }
  default       { $nothing = 1; }
}

# Code issu de la documentation de Sébastien Aperghis-Tramoni
given ($number) {
  when (42) { say "La Réponse"; }
  when (56) { say "fer standard"; }
}

use Regexp::Common qw/net/;

given ($host) {
  when ("localhost")        { say "adresse locale"; }
  when (/$RE{net}{IPv4}/)   { say "adresse IPv4"; }
  when (/$RE{net}{domain}/) { say "FQDN"; }
  default                   { say "type d'argument inconnu"; }
}
```

- **un when réussi termine le given correspondant ;**
- **un break permet de sortir du given ;**
- **un continue permet d'aller au when suivant ;**
- **default est lu si rien d'autre ne correspond.**

Dans le core de Perl 5.10 ou plus, le module Switch existe et s'utilise de la façon suivante :

```
use Switch;

switch ($string) {
  case /^abc/ { $abc     = 1; }
  case /^def/ { $def     = 1; }
  case /^xyz/ { $xyz     = 1; }
  else        { $nothing = 1; }
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
}
```

Répétitions et itérations

Auteurs : 2Eurocents ,

Plusieurs formes de parcours de boucles, de répétitions ou d'itérations sont disponibles. Elles sont à adapter en fonction des besoins. Le but est toujours de parcourir un bloc de code tant que certaines conditions sont vérifiées, ou bien un certain nombre de fois, ou bien pour toutes les valeurs d'une liste, comme les fonctions map et grep.

Boucles bornées pour programmeurs obstinés

Auteurs : 2Eurocents ,

Ce sont les boucles qui exigent une condition de sortie. On y trouve les boucles avec une condition permanente :

```
while (condition)
  {
    # bloc d'instructions
  }
```

où le bloc d'instructions est exécuté tant que la condition est vraie. On y trouve aussi les boucles avec une condition terminale :

```
until (condition)
  {
    # bloc d'instructions
  }
```

où le bloc d'instructions est exécuté jusqu'à ce que la condition soit vraie.
Ces deux formes de boucle acceptent aussi une syntaxe abrégée lorsque le bloc d'instructions se réduit à une seule instruction :

```
instruction while (condition);
```

ou

```
instruction until (condition);
```

Boucles comptées, mais quand on aime...

Auteurs : 2Eurocents ,

Ce sont des boucles pour lesquelles une variable sert à dénombrer les itérations et à arrêter la répétition lorsqu'une valeur donnée est atteinte. La syntaxe est alors :

```
for (initialisation ; condition ; incrementation)
{
  # bloc d'instructions
}
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

où l'initialisation est l'affectation de valeurs initiales aux variables d'index de boucle ; la condition est la condition d'arrêt de la répétition et l'incrémentation est l'instruction qui altère les variables d'index de boucle à chaque passage. Chacune des trois clauses est optionnelle. L'initialisation peut être faite en dehors de la boucle, la condition et l'incrémentation pouvant être réalisées dans le bloc d'instructions. Par exemple :

```
for ($i=0 ; $i<10 ; $i++) {
  print "$i\n";
}
```

affiche les valeurs de 0 à 9.

```
for (($i, $j)=(0, 10) ; $i<10 ; ($i++,$j--)) {
  print "[$i,$j]\n";
}
```

affiche un $i croissant et un $j décroissant. En fait, la syntaxe est équivalente à un :

```
initialisation
while (condition) {
  # bloc d'instructions
  incrémentation
}
```

Boucles d'énumération

Auteurs : 2Eurocents ,

Il est possible de parcourir la totalité des éléments d'une liste, et pas seulement au moyen des instructions map et grep. La fonction each() est destinée aux énumérations de tableaux associatifs. Elle prend une table de hachage comme argument et retourne une clef de cette table. A l'appel suivant, elle retournera la clef suivante. Et ainsi de suite jusqu'à épuisement des clefs. On voit bien qu'intégrée à une boucle while/until ou for, cette fonction peut énumérer tous les éléments d'un hachage. L'autre construction d'énumération très utilisée est la boucle foreach. La syntaxe complète est :

```
foreach $variable ( @liste ) {
  # bloc d'instructions
}
```

Le bloc d'instructions est exécuté pour chaque valeur contenue dans la liste @liste. La variable $variable prendra, à chaque itération, les valeurs successives des éléments de @liste. Il existe aussi une syntaxe abrégée utilisant la variable implicite de contexte $_. Cette variable, interne à Perl permet de simplifier de nombreuses syntaxes. La variable $_ contient, le plus souvent, l'élément en cours de traitement - d'où l'appellation de variable de contexte :

```
foreach ( @liste ) {
  # bloc d'instructions
  # l'élément en cours est $_
}
```

Un petit exemple supplémentaire, sachant que la fonction rand retourne un nombre flottant pseudo aléatoire dans l'intervalle [0..n], n étant passé en paramètre :

```
# liste des sentences pour chaque pétale possible (6 x 10)
my @marguerite = ( 'je t\'aime', 'un peu', 'beaucoup', 'passionnément', 'à la folie', 'pas du tout'
 ) x 10;

# Tirage du nombre de pétales, de 0 à 59
my $i = int( rand(59) );
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
# Pour tous les pétales...
foreach ( 0 .. $i ) {

  # On effeuille la marguerite
  print $marguerite[$_] . "\n";
}
```

Points de ruptures... de séquences

Auteurs : 2Eurocents ,

Un programme avec un bon algorithme se déroule normalement sans problème. Tous les cas sont prévus dans les alternatives, les boucles sont bien conçues et tout est pour le mieux. Cependant, il arrive que des changements dans les traitements, au cours de la maintenance du code, ou bien des difficultés imprévues de conception, obligent le programmeur à introduire des conditions particulières de rupture des structures de contrôle. Ces ruptures ne sont pas un signe de bonne conception... plutôt le signe qu'une modification, apparemment bénigne qui aurait eu de grosses répercussions sur la structure du code et qu'il a été préférable d'en limiter l'ampleur par un petit traitement localisé.

- **redo est l'instruction de rupture qui permet de reprendre l'itération en cours au début du bloc de code ;**
- **next est l'instruction qui permet de passer à l'itération suivante. Elle effectue un saut direct à la fin du bloc de code ;**
- **last est l'instruction qui permet d'arriver directement à la fin de la dernière itération. On sort des traitements/ boucles et on continue l'exécution du programme aux instructions qui suivent.**

Ces ruptures de séquence fonctionnent aussi sur des boucles imbriquées. Dans ce cas, pour savoir de quelle boucle on souhaite rompre la séquence, il faut étiqueter les boucles et préciser l'étiquette à la rupture de séquence :

```
EXTERNE: foreach $i ( 0 .. 10 ) {
  INTERNE: foreach $j ( 0 .. 10 ) {
      next EXTERNE if ( ( $i + $j ) > 10 );
      print "i=$i - j=$j\n";
  }
}
```

Dès que la valeur de $j est suffisante pour que ($i+$j) soit supérieur à 10, on passe à la valeur suivante de $i, par passage à l'itération suivante de la boucle EXTERNE. On aurait aussi pu remplacer le next EXTERNE par un last INTERNE, le résultat aurait été le même... mais en Perl, il y a plus d'une façon de le faire !

Sauter, c'est le meilleur moyen de se casser la figure !

Auteurs : 2Eurocents ,

Il existe aussi une instruction goto qui permet de faire un saut dans l'exécution du code, jusqu'à une étiquette donnée (fixée à l'écriture du programme, ou évaluée dynamiquement. Utiliser ce type d'instruction n'est pas vraiment une

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

bonne idée. Le langage étant par ailleurs riche en structures de contrôles et en instructions de ruptures, l'usage de cette instruction peut être carrément néfaste, à la maintenance du code si ce n'est à son exécution.

L'appel des fonctions à la pelle !

Auteurs : 2Eurocents ,

Puisque les sauts sont déconseillés dans le contrôle du flux d'exécution du programme, il existe une structure "idoine" qui permet d'abandonner le bloc courant pour exécuter un autre bloc de code et revenir ensuite à notre point de "digression". C'est la notion de sous-routine, procédure ou fonction. Cela permet d'écrire une seule fois un bloc de code qui sera exécuté à de nombreuses reprises dans un programme, depuis plusieurs endroits différents. Une fonction se déclare avec le mot-clef "sub", suivi du nom de la fonction, suivi du bloc de code à exécuter :

```
sub a_la_trace {
  print "Tout passage par la fonction est suivi a_la_trace\n";
}
```

Cette fonction s'appelle simplement par son nom, dans tout le code :

```
print "Début";
a_la_trace();
print "Milieu";
a_la_trace();
print "Fin";
```

Le départ, des paramètres à mettre

Auteurs : 2Eurocents ,

La fonction ainsi définie peut aussi admettre des paramètres. Perl n'étant pas très strict ni rigide, il peut même laisser le nombre de paramètres des fonctions totalement libre. Une fonction qui admet des paramètres s'appelle en précisant les paramètres entre parenthèses après le nom de la fonction :

```
print "Début";
A_la_trace("Début", "Milieu");
print "Milieu";
A_la_trace("Milieu", "Fin");
print "Fin";
```

réalise l'appel de A_la_trace en lui passant deux chaînes comme paramètres à chaque fois. Les parenthèses ne sont toutefois pas une obligation, mais elles aident grandement à la lisibilité du code et permettent d'éviter quelques soucis avec la précédence des opérateurs...

Et on les récupére comment, les paramètres dans la fonction ?

Auteurs : 2Eurocents ,

Lors de l'appel à une fonction, Perl range les paramètres dans une liste spéciale, "@_", interne à la fonction. Cette liste se comporte comme toutes les autres. Il n'est, par contre, pas recommandé de tenter de la modifier. Il est donc possible de récupérer les valeurs de différentes façons :

```
my ($param1, $param2) = @_;
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

récupére les deux premiers paramètres (@_ est inchangée).

```
my ($param1) = @_;
```

est équivalent à my $param1 = $_[0];

```
my $param = shift;
```

récupére le premier paramètre de la liste et le supprime d'icelle. Le nombre de paramètres effectif étant laissé libre, il peut être pratique de le connaître, ne serait-ce que pour effectuer le nombre de shift correct pour récupérer toutes les valeurs. Le nombre de paramètres reçus se trouve simplement par l'évaluation de @_ en contexte scalaire.

```
my $nombre_arguments = @_;
# ou bien
my $nombre_arguments = scalar @_;
```

Et avec des listes comme paramètres ?

Auteurs : 2Eurocents ,

On l'a vu, à la réception dans la fonction, les paramètres sont rangés dans une liste. Si lors de l'appel on a spécifié une liste comme paramètre de la fonction, celle-ci va s'aplatir dans la liste @_. Pour conserver le caractère "liste" des paramètres, il faut passer par la notion de référence, qui sera vue ultérieurement. De même, si l'on souhaite modifier la valeur des paramètres, les références pourront nous aider.

Pensons au retour

Auteurs : 2Eurocents ,

Une fonction se termine dans deux cas :

- **la fin du bloc de code de la fonction est atteinte. Retour à l'appelant ;**
- **une instruction return est rencontrée. Retour à l'appelant, mais on lui transmet les valeurs qui suivent le mot-clef return.**

Les valeurs retournées peuvent être aussi bien un scalaire qu'une liste. Il est donc possible d'avoir des fonctions retournant des listes d'éléments (concept que l'on a finalement déjà rencontré avec map ou grep).
Nous sommes maintenant à peine aguerris mais suffisamment équipés pour aller beaucoup plus loin avec le langage Perl... l'odyssée ne fait que commencer !

La fonction AUTOLOAD

Auteurs : Woufeil ,

Perl vous permet de définir une fonction particulière qui sera appelée si l'utilisateur tente d'invoquer une fonction non définie dans le paquetage (l'espace de nom). Cette fonction particulière s'appelle AUTOLOAD. Si elle est appelée, une variable globale au paquetage nommée $AUTOLOAD, qui contiendra le nom de la fonction que l'on a essayé d'appeler, sera créée. Attention, pour utiliser cette variable, il faut la déclarer avec our, on ne peut pas l'utiliser sans la déclarer. Par exemple, si l'on veut signaler que la fonction appelée par l'utilisateur n'existe pas sans pour autant faire boguer le programme, on peut le faire ainsi :

```
sub AUTOLOAD {
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
 our $AUTOLOAD;
 print "La fonction $AUTOLOAD n'a pas été définie !";
}
```

Plus utile, si on a un module assez gros que l'on est susceptible d'utiliser dans un programme sans être sûr de le faire, on peut demander à la fonction AUTOLOAD de le faire via require (et non use, ce dernier étant réglé à la compilation et non à l'exécution comme require). Si l'utilisateur tente d'appeler une fonction du module et que ce dernier n'a pas été inclus, la fonction AUTOLOAD l'inclura et appellera la fonction en question sans que l'utilisateur n'ait quoi que ce soit à faire.

Un Shell en six lignes !

Auteurs : Woufeil ,

La fonction AUTOLOAD permet de créer votre propre Shell en quelques lignes :

```
sub AUTOLOAD {
 my $fonc = our $AUTOLOAD;
 $fonc =~ s/.*::///; #on enlève le nom du paquetage et les ::
 system($fonc, @_) #appelle $fonc avec la liste d'arguments passée à AUTOLOAD
}
cd (mount) #appelle la fonction système cd en lui passant mount en paramètre
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

L'entrée standard

Auteurs : GLDavid , Djibril ,

Au cours de vos programmes, vous aurez sans doute besoin de paramètres propres à l'utilisateur. Celui-ci aura alors à renseigner au programme ses propres données. Cette interaction, dans sa forme la plus simple, s'effectue via l'entrée standard (le clavier). Pour avoir accès à ce qui est rentré via le clavier, vous aurez besoin de manipuler l'opérateur STDIN :

```
print "Rentres une phrase :\n";
$in = <STDIN>;
chomp $in;
print "$in\n";
```

Dans ce programme très simple, l'utilisateur devra rentrer une phrase ou un chiffre qui sera validé par l'appui sur la touche *Entrée*. Supposons que l'utilisateur ait rentré le chiffre 2. La variable scalaire contient ainsi pour le moment "2\n". Le caractère de retour chariot (la touche entrée) ne nous intéresse pas. D'où l'appel de la fonction chomp qui enlève le dernier caractère s'il s'agit d'une fin de ligne. Le programme affiche enfin le contenu de la variable scalaire $in. chomp supprime le dernier caractère d'une chaîne si ce dernier est un retour chariot et elle retourne ce caractère.
Voici quelques explications d'utilisation de la fonction chomp :

```
my $chaine = "Bonjour\n";
my $chomp  = chomp $chaine;    # $chaine = "Bonjour",  $chomp = "\n"

my $chaine = "tutu\ntoto";
my $chomp  = chomp $chaine;    # $chaine = "tutu\ntoto",  $chomp1 = ""

my $chaine = "\n";
my $chomp  = chomp $chaine;    # $chaine = "",  $chomp1 = ""

my $chaine = "123";
my $chomp  = chomp $chaine;    # $chaine = "123",  $chomp1 = ""
```

Un diamant dans Perl

Auteurs : GLDavid ,

Nous avons vu tout à l'heure que l'opérateur (ou handle, cf. les fichiers) STDIN était contenu dans un opérateur dit opérateur diamant <>. Cet opérateur est très utile notamment pour la manipulation des arguments fournis à votre programme. Un argument, pour un programme Perl, désigne un fichier que vous passez en paramètre à votre programme. Votre code lira ainsi le fichier donné :

```
perl mon_programme.pl GLDavid Djibril gnux 2eurocents
```

Notre programme Perl va devoir lire chacun des fichiers passés en argument et afficher leur contenu sur la sortie standard :

```
foreach $ligne (<>) {
 chomp $ligne;
 print "$ligne\n";
}
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Dans notre boucle, nous lisons chacun des fichiers passés séquentiellement. L'opérateur chomp le dernier caractère si ce dernier est un retour chariot et nous affichons sur la sortie standard la ligne en cours du fichier.

Ecrire sur la sortie standard

Auteurs : GLDavid ,

La sortie standard détermine le média de sortie avec lequel votre programme affichera une information à vos utilisateurs. De manière commune, la sortie standard sera votre écran. Dans nos précédents codes sources, nous avons vu que pour afficher quelque chose, nous avions recours à la fonction print. Cette fonction prend en argument une chaîne de caractères qu'elle affichera sur la sortie standard. Mais elle permet aussi de vous afficher le contenu de variables.

```
$out = "Bonjour Maître.\n";
print "$out\n";
print "Voici mes maîtres :\n";
@tableau = qw /GLDavid Gnux Djibril 2Eurocents/;
print "@tableau";
```

Enfin, dans votre chaîne de caractères fournie à print, rien ne vous empêche de procéder à des concaténations.

Comment formater la sortie standard ?

Auteurs : GLDavid ,

Chers pratiquants du C, n'ayez pas peur de Perl ! Vous y retrouverez la première fonction que vous avez utilisée à vos débuts : printf ! Pour les non Cistes (codeurs en C), printf est une fonction permettant d'afficher, certes, sur la sortie standard mais aussi de formater ce que l'on souhaite imprimer. Prenons l'exemple d'un nombre avec décimales. Naturellement, pour certains de ces nombres, on ne veut retenir que quelques chiffres après la partie entière :

```
my $taux = 6.55957;
print "Rentre une valeur en euros : ";
my $euro = <STDIN>;
chomp $euro;
my $franc = $euro * $taux;
printf "%.3f euros vaut %.3f francs.\n", $euro, $franc;
```

```
Rentrez une valeur en euros : 12.2
12.200 euros vaut 80.027 francs.
```

**Dans cet exemple, nous demandons à l'utilisateur de rentrer un montant correspondant à une valeur en euros.
Au préalable, nous aurons défini le taux de change dans la variable scalaire $taux. La conversion s'effectue et nous affectons à la variable $franc le résultat de la conversion. Notre ligne la plus intéressante est la dernière. Plutôt que d'appeler print, nous voulons seulement retenir trois décimales pour chacune de ces valeurs ($euro et $franc).
A l'aide du drapeau %.3f, nous indiquons à printf que pour les variables $euro et $franc nous ne souhaitons afficher que la partie entière ainsi que les 3 chiffres suivants le séparateur décimal. Notez enfin que vous aurez à déterminer pour chaque variable un drapeau correspondant. Quelques drapeaux :**

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Drapeau	Signification
%d	Affiche un entier
%g	Notation automatique décimale, entière ou exponentielle
%.3f	Affiche un flottant avec 3 chiffres après le séparateur décimal
%x	Affiche un hexadécimal
%o	Affiche un octal
%s	Affiche une chaîne de caractères
%10s	Affiche une chaîne de caractères avec une justification de 10 espaces vers la droite
%-10s	Affiche une chaîne de caractère avec une justification de 10 espaces vers la gauche

Voir : perldoc perlfunc printf

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > S'initier à Perl > Expressions régulières

Définition d'une expression régulière

Auteurs : Djibril ,

C'est un motif (pattern) dont on recherche la présence (matching) dans les chaines de caractères. Cette recherche permet de tester, filtrer, ou effectuer des remplacements de données (motifs, sous-motifs...). Une expression régulière est un "motif" de recherche constitué d'un caractère ou ensemble de caractères. L'opérateur conditionnel est "=~" et signifie "ressemble à".

- **Syntaxe : $chaine =~ /mon motif/; : la variable $chaine ressemble à l'expression "mon motif" ;**
- **Négation: $chaine !~ /mon motif/; : la variable $chaine ne ressemble pas à l'expression "mon motif".**

Exemple : Je cherche à vérifier si ma chaîne de caractères "Djibril" commence par le motif "dji".

```
#!/usr/bin/perl
use warnings;
use strict;
use Carp;

my $chaine = "Djibril";
if ( $chaine =~ /^dji/ ) {
  print " ok\n ";    # => ok
}
```

Remarque : Par défaut l'expression régulière est appliquée à la variable $_.

```
if ( $_ =~ /exp/ )
# Identique à
if ( /exp/ )
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > S'initier à Perl > Expressions régulières > Classes de caractères

Un caractère et ensemble de caractères

Auteurs : Djibril ,

- **tout caractère alphabétique en minuscule: [a-z] ;**
- **tout caractère alphabétique en majuscule : [A-Z] (il existe des options pour prendre en compte la casse, voir plus loin) ;**
- **tout caractère numérique : [0-9] ;**
- **tout caractère alphanumérique : [a-zA-Z0-9].**

Caractères spéciaux et prédéfinis

Auteurs : Djibril ,

Notation	Signification	Equivalent
\n	Un retour chariot.	
\t	Une tabulation.	
^	Le caractère spécial "^" a deux significations différentes : Dans un ensemble [..], il signifie "tout sauf" En dehors il signifie "ligne commence par"	[^0-9] /^dji/
\w	Un caractère alphanumérique, avec le "souligné" couramment appelé "underscore" (_).	[a-zA-Z0-9_]
\W	Tout sauf un caractère alphanumérique et le souligné (_) compris.	[^a-zA-Z0-9_]
\d	un caractère numérique.	[0-9]
\D	Tout sauf un caractère numérique.	[^0-9]
\s	Un espace.	
\S	Tout sauf un espace.	
$	Fin de ligne.	/son$/ expression se terminant par "son"
$&	Motif qui correspond à l'expression régulière.	/motif/ (exemple ci-dessous)
$`	Sous-ensemble qui se trouve avant $&.	(exemple ci-dessous)
$'	Sous-ensemble qui se trouve après $&.	(exemple ci-dessous)

Voici un exemple de script utilisant différents caractères :

```
#!/usr/bin/perl
use warnings;
use strict;
use Carp;
my $mon_texte = "Il fera beau d'ici 15 heures 30";
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
if ( $mon_texte =~ /fera/ ) {
    print "$&\n";  # $& = "fera"
    print "$`\n"; # $` = "Il ", attention, l'espace est aussi récupéré
    print "$'\n";       # $' = " beau d'ici 15 heures 30"
}
if ( $mon_texte =~ /\d$/ ) {
    print "$mon_texte\n";  # Il fera beau d'ici 15 heures 30
}
if ( $mon_texte =~ /[0-9]$/ ) { # Ecriture semblable à la précédente
    print "$mon_texte\n";  # Il fera beau d'ici 15 heures 30
}
if ( $mon_texte =~ /[a-z]$/ ) {
    print "OK\n";
}
else {
    print "Pas de bol
\n"; # Pas de bol (car ma chaîne ne se termine pas par un caractère alphabétique)
}
```

Métacaractères et quantificateurs

Auteurs : Djibril ,

Il s'agit maintenant d'étudier "le langage" des expressions régulières, fondé sur le rôle d'opérateur des métacaractères. Rappelons qu'il faut les "échapper" avec "\" pour neutraliser leur action d'opérateurs.

Notation	Signification
.	Représente un caractère quelconque, sauf \n (comportement par défaut, modifiable).
*	Marque la possible répétition du caractère précédent (ou de l'expression précédente entre parenthèses) 0 ou n fois.
+	Marque la répétition du caractère précédent (ou de l'expression précédente entre parenthèses) au MOINS 1 fois.
?	Marque la possible répétition du caractère précédent (ou de l'expression précédente entre parenthèses) 0 ou 1 fois.
[..]	Recherche L'UN des caractères de l'ensemble des caractères entre crochets.
[^..]	Recherche tout sauf les caractères qui sont entre les crochets.
^	La recherche s'effectue en début de chaîne ex : /^regex/.
$	La recherche s'effectue en fin de chaîne ex : /regex$/.
\	Annule le rôle de métacaractère du caractère qui suit, et lui permet d'avoir sa signification usuelle.
|	Joue le rôle de "ou" entre 2 expressions. (ex : /toto|titi/) => toto ou titi.
(...)	Rôle de groupement et de mémorisation de la regex comprise.
{n,m}	Le nombre de répétitions attendu va de n à m (et porte sur le caractère ou

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

	l'expression précédent), m et n inférieurs à 65536.
{n}	**Le nombre de répétitions attendu doit être exactement égal à n.**
{n,}	**Le nombre de répétitions attendu est au moins n.**
{,m}	**Le nombre de répétitions attendu est au plus m.**
	{0,} {1,} {0,1} équivalent respectivement à *, + et ?

N.B. : les Métacaractères sont : " ^ | { } [ ] ( ) / \ $ + * ? ." Exemple sur l'utilisation du "\":

```
#!/usr/bin/perl
use warnings;
use strict;
use Carp;
my $chaine = "La/les maison(s) en Normandie ";

# Pour empêcher que Perl interprète "/ ( )" dans ma chaîne comme des métacaractères,
# on utilise le "\"
if ( $chaine =~ /^La\/les maison\(s\)/ ) {
  print " La ligne est $chaine ";    # => La/les maison(s) de Normandie
}
```

Il existe une méthode beaucoup plus propre et lisible pour la manipulation des métacaractères. Elle consiste à utiliser la fonction Perl quotemeta() qui protège tous les métacaractères d'une chaîne. Exemple :

```
#!/usr/bin/perl
use warnings;
use strict;

my $chaine  = "La/les maison(s) en Normandie coûte 200 000 euros";
my $pattern = "La/les maison(s)";
$pattern = quotemeta($pattern);

if ( $chaine =~ /^$pattern/ ) {
  print " la ligne est $chaine ";
}
```

résultat quotemeta

```
mon pattern : La\/les\ maison\(s\)
 la ligne est La/les maison(s) en Normandie coûte 200 000 euros
```

Caractères d'ancrages

Auteurs : Djibril ,

Ce sont des symboles dont la présence indique une contrainte de positionnement du motif à la recherche dans le texte. Nous les avons déjà étudiés ci-dessus.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

^pipo	**La recherche du motif pipo doit être effectuée uniquement en début de chaîne.**
regex$	**recherche du motif regex doit être effectuée uniquement en fin de chaîne.**

Mémorisation (parenthèses de regroupement et de capture)

Auteurs : Djibril ,

La présence de parenthèses autour d'une partie d'un motif (qu'on appellera sous-motif) est nécessaire pour plusieurs raisons et objectifs :

- **pour marquer un regroupement de plusieurs caractères sur lesquels on veut agir, le plus souvent, par la pose d'un quantificateur. Pensez à la différence entre les "mots" papa+ et pa(pa)+ ;**
- **pour signifier le besoin de mémoriser la sous-chaîne du texte étudié qui satisfait le sous-motif. On parle pour cette raison de parenthèses de capture. On peut référencer le sous-motif par $1 à $n ($n est une variable spéciale : le contenu de la n-ième parenthèse).**

Exemple :

```
#!/usr/bin/perl
use warnings;
use strict;

my $chaine = "les maisons valent 5000000 euros";
if ( $chaine =~ /\s(\d+\seuros)$/ ) {
  my $prix = $1;
  print "il coute donc $prix\n";    #il coute donc 5000000 euros
}
```

Une autre façon :

```
#!/usr/bin/perl
use warnings;
use strict;

my $chaine = "les maisons valent 5000000 euros";
my ($prix, $monnaie) = $chaine =~ /\s(\d+)\s+(euros)$/;
if ( defined $prix ) {
  print "Il coute donc $prix $monnaie\n";    # Il coute donc 5000000 euros
}
```

Dans le deuxième exemple ci-dessus, il est important de ne pas oublier les parenthèses "($prix,$monnaie)" car les motifs $1, $2... $n sont renvoyés sous forme de liste.

Il existe une autre façon de capturer depuis perl 5.10 : captures nommées. Voici un exemple :

```
#!/usr/bin/perl
use warnings;
use strict;

my $chaine = "les maisons valent 5000000 euros";
if ( $chaine =~ /\s(?<prix>\d+)\s+(?<monnaie>euros)$/ ) {
  print "Il coute donc $+{prix} $+{monnaie}\n";    # Il coute donc 5000000 euros
}

foreach my $nom_capture ( keys %+ ) {
  print "$nom_capture : ", $+{$nom_capture},"\n";
}
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
Il coute donc 5000000 euros
monnaie : euros
prix : 5000000
```

C'est assez pratique. Les captures se retrouvent dans la table de hachage spéciale %+.

Les modificateurs

Auteurs : Djibril ,

Une expression régulière peut être suivie d'options cumulables, que l'on appelle modificateurs. Ils modifient l'interprétation d'une expression régulière.

Syntaxe

```
if ( $toto =~ /expr/msogix ) {
  # ...
}
```

Notation	Signification
i	Recherche insensible à la casse (majuscules/ minuscules).
g	La recherche est globale, de toutes les occurrences.
x	Les espaces et sauts de lignes sont ignorés. le caractère # permet de faire un commentaire (pensez à l'utiliser avec les séparateurs {}{}.
m	Traiter la chaîne comme des lignes multiples (^ et $ s'appliqueront à chaque ligne).
s	Traiter la chaîne comme une ligne simple (le caractère . reconnaît aussi les sauts de ligne).
o	Compilation du motif uniquement la première fois.

Exemple de script utilisant ces modificateurs :

```
#!/usr/bin/perl
use warnings;
use strict;

my $chaine = "atgccATCccca";
if ( $chaine =~ /[atgcATGC]/ ) {
  print "OK sans utiliser l'option i\n";    #=> OK sans utiliser l'option i
}
if ( $chaine =~ m/[atgc]/i ) {
  print "OK avec utiliser l'option i\n";    #=> OK avec utiliser l'option i
}
if ( $chaine =~ /(atc)/gi ) {
  print "motif $1 trouvé\n";               #=> motif ATC trouvé
}

my $mon_texte = "il fait jour\nIl fait nuit\nBonsoir";
if ( $mon_texte =~ /jour$/ ) {
  print "$mon_texte\n";                    #=> rien
}
if ( $mon_texte =~ /jour$/m ) {
  print "$mon_texte\n";                    #=> il fait jour
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
  #   Il fait nuit
  #   Bonsoir
}
```

Voici un exemple d'utilisation des modificateurs msx (utilisation récommandée), ainsi que l'utilisation des séparateurs {}

```
#!/usr/bin/perl
use warnings;
use strict;

my $chaine = "les maisons valent 5000000 euros\nl'hôtel coute 100000 dollars\n";
my (%data) = $chaine =~ m{       # m => signifie match
                         ^[^\n]+  # ligne commence par tout sauf un \n jusqu'à l'espace
                          \s      # Reconnait un espace
                          (\d+)   # capture un nombre
                          \s*     # suivi de plusieurs espaces ou non
                          (\w+)   # suivi d'un mot capturé
                          $       # Fin d'une ligne
                           }gmxs;    # m => multiligne
                                     # x permet d'ignorer les blancs et les \n et
                                     # les commentaires via #
                                     # s => le point reconnait maintenant le \n
                                     # g => plusieurs occurrences
if (%data) {
  foreach ( keys %data ) {
    print "$_ : $data{$_}\n";
  }
}
```

Remplacement et substitution

Auteurs : Djibril ,

Comme le fait la commande "sed" sous Unix, Perl permet de faire des remplacements sur une chaîne de caractères en utilisant

Syntaxe

```
$chaine =~ s/motif/remplacement/;
```

```
#!/usr/bin/perl
use warnings;
use strict;
my @fruits = qw (banane pomme melon);

foreach (@fruits) {
  s/e$/es/;
}
print "@fruits\n";    #bananes pommes melon

my $tel = "01.12.22.44.99";
$tel =~ s{
         0         # le 0 sera remplacé
         ([^\n]+)  # numero à capturer
          }
         {\+(00\.33) $1}xms;
print "$tel\n";       #+(00.33) 1.12.22.44.99
my $arbres = "manguier pommier cerisier";
$arbres =~ s/(\w+)ier/arbres à $1es,/g;
print "list: $arbres\n";    #=> list: arbres à mangues, arbres à pommes, arbres à cerises,
$arbres =~ s/,$/./;
print "list: $arbres\n";    #=> list: arbres à mangues, arbres à pommes, arbres à cerises.
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

tr	**Cette commande permet de transformer une chaîne en une autre chaîne ou de modifier des occurrences dans une chaîne.**
split	**Découpe et range dans une liste les éléments trouvés contenus entre un délimiteur défini. split(délimiteur,$variable, $option);, $option est facultatif.**
join	**Concatène une valeur à chaque élément d'une liste et range le tout dans une variable. join($valeur,@list);**

```perl
# tr
$name =~ tr/A-Z/a-z/;    # Transforme toutes les majuscules en minuscules
$name =~ tr/er/ab/;      # Transforme les e en a et les r en b

# split
$a = "1:2:3:4::5";
@list = split( /:/, $a );    #=> ("1","2","3","4","","5")

# join
$valeur = "+";
@list   = ( "1", "2", "3", "4" );
$res    = join( $valeur, @list );    #=> ("1+2+3+4+")
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > Perl avancé

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > Perl avancé > Les fichiers

Notion de handle de fichier

Auteurs : GLDavid , Djibril ,

Dans les précédentes parties, vous avez conçu vos programmes Perl pour que ceux-ci puissent converser via les entrées/sorties (E/S) standard à savoir : STDIN (le clavier), STDOUT (la console) et STDERR (sortie standard en cas d'erreur). Dans ce chapitre, nous allons maintenant apprendre à lire à partir des fichiers stockés sur un média (disque dur, CDROM, disquette...). Le langage Perl nécessite pour la lecture/écriture de fichiers des handles.
Un handle n'est qu'une connexion entre votre programme Perl et le fichier à lire ou à écrire. Il existe déjà six handles de fichiers réservés : STDIN (soit le flux d'entrée standard), STDOUT (flux de sortie standard), STDERR (direction des erreurs), DATA, ARGV et ARGVOUT.
Vous lirez dans certaines documentations le mot "file handle". Il correspond tout simplement à un identifiant du fichier après ouverture, une sorte de descripteur du fichier.

Ouvrir et fermer un fichier

Auteurs : GLDavid ,

Rien ne vous empêche dans Perl de créer et nommer vos propres handles. De même, les connaisseurs d'Unix seront sans doute ravis de savoir que les redirections d'entrées/sorties standards sont primordiales pour indiquer au handle la direction du flux. Voici d'ailleurs un exemple pour illustrer l'ouverture d'un fichier en Perl. Nous supposerons qu'il existe dans le même répertoire que notre script un fichier "toto.txt" :

```
open TOTO, "<toto.txt ";
```

Dans cet exemple, le chevron "<" indique que nous sommes en lecture. Bien naturellement, il est envisageable de mettre dans une variable scalaire le chemin de votre fichier :

```
$chemin = "/home/gldavid/toto.txt";
open TOTO, "<$chemin";
```

L'écriture reprend la même syntaxe, sauf que vous aurez deux méthodes :

```
open TOTO, ">toto.txt";
open TATA, ">>tata.txt";
```

Quelle est la différence entre ces deux instructions ? Les amateurs d'Unix vous diront que dans le premier cas, nous écrivons sur toto.txt. Si celui-ci n'existe pas, Perl le crée avant d'écrire dans ce fichier. Dans le cas où celui-ci existe déjà, le contenu sera écrasé par le traitement effectué.
Dans le deuxième cas, si le fichier tata.txt n'existe pas, Perl le crée puis procède à l'écriture. Dans le cas où le fichier existe au préalable, Perl écrit à la suite du contenu.
Enfin, il s'agit de procéder proprement.
Comme dans tous les langages, dès que vous ouvrez un flux, pensez à le fermer !

```
open TOTO, ">>toto.txt";
#Je lis le fichier toto.txt et j'exécute mon traitement
close(TOTO);
```

L'instruction close sur un handle permet de fermer tout lien entre votre processus Perl et le fichier visé.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Depuis Perl 5.6, il y a une meilleure façon d'ouvrir un fichier, veuillez lire les sections suivantes.

Vivre et laisser mourir !

Auteurs : GLDavid ,

Naturellement, il se peut que vous vous trompiez dans le chemin de votre fichier. Pour le cas où vous souhaitez gérer de telles exceptions, Perl vous offre le choix de tuer votre application et de pouvoir disposer d'un message d'erreur. La close die produit un tel message :

```
open FILE, "< toto.txt" or die "toto.txt n'existe pas !\n";
while ($ligne = <FILE>) {
print "$ligne\n";
}
```

Lorsque votre script Perl arrive à cette instruction et si celui-ci ne trouve pas le fichier toto.txt, la close die entre en jeu. Votre programme s'arrêtera en vous affichant le message *"toto.txt n'existe pas !"*. Vous pourrez bien sûr utiliser aussi la variable $! qui marque l'erreur émanant du système. Dans le cas où tout se passe bien à l'ouverture du handle, le programme affichera sur la sortie standard le contenu du fichier toto.txt. Mais, vous pouvez aussi faire en sorte d'afficher votre message d'erreur sans pour autant arrêter votre programme Perl. En utilisant la close warn, vous produirez le message d'erreur de votre choix sans que le script ne s'arrête :

```
open FILE, "< toto.txt" or warn "toto.txt n'existe pas !\n";
```

Tests sur les fichiers

Auteurs : Djibril ,

Perl vous autorise à avoir accès au moindre renseignement de vos fichiers. Le tableau suivant résume les différents tests possibles en Perl sur les fichiers :

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Syntaxe	Signification	Remarque
-r	Le fichier est lisible par l'uid et le gid effectifs	Spécifique à Unix
-w	Le fichier peut être écrit par l'uid et le gid effectifs	Spécifique à Unix
-x	Le fichier peut être exécuté par l'uid et le gid effectifs	Spécifique à Unix
-R	Le fichier est lisible par l'uid et le gid réels	Spécifique à Unix
-W	Le fichier peut être écrit par l'uid et le gid réels	Spécifique à Unix
-X	Le fichier peut être exécuté par l'uid et le gid réels	Spécifique à Unix
-o	Le fichier appartient à l'uid effectif	Spécifique à Unix
-O	Le fichier appartient à l'uid réel	Spécifique à Unix
-e	Le fichier existe	
-z	Le fichier existe et a une taille de 0	
-s	Le fichier existe et a une taille différente de 0 (retourne la taille)	
-f	Le fichier est un fichier ordinaire	Spécifique à Unix
-d	L'entrée est un répertoire	
-l	L'entrée est un lien symbolique	Spécifique à Unix
-S	L'entrée est une socket	Spécifique à Unix
-p	L'entrée est un pipe nommé FIFO	Spécifique à Unix
-b	L'entrée est un fichier spécial de blocs (comme un périphérique montable)	Spécifique à Unix
-c	L'entrée est un fichier spécial de caractères (périphérique E/S)	Spécifique à Unix
-u	Le bit setuid est activé pour ce fichier	Spécifique à Unix
-g	Le bit setgid est activé pour ce fichier	Spécifique à Unix
-k	Le sticky bit est activé pour ce fichier	Spécifique à Unix
-t	Le handle de fichier (défaut STDIN) est ouvert sur une tty	Spécifique à Unix
-T	Le fichier est un fichier texte. Retourne vrai pour un fichier vide ou pour un handle de fichier en fin de fichier (EOF)	
-B	Le fichier est un fichier non texte (binaire). Retourne vrai pour un fichier vide ou pour	

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

	un handle de fichier en fin de fichier (EOF)	
-M	**Date de la dernière modification exprimée en jours (avec décimale). La valeur retournée correspond à l'âge de fichier au moment du démarrage du programme**	
-A	**Date de la d'accès exprimée en jours (avec décimale). La valeur retournée correspond à l'âge de fichier au moment du démarrage du programme**	
-C	**Date de la de changement exprimée en jours (avec décimale). La valeur retournée correspond à l'âge de fichier au moment du démarrage du programme**	

Apprenez-le par coeur ! Interro surprise la semaine prochaine ! A l'aide de ces tests, on peut, par exemple, tester l'existence d'un fichier avant de l'ouvrir :

```
$file = "/home/gldavid/toto.txt";
if ( -e $file ) {
  OPEN FILE, "< $file" or die "Un problème est survenu pendant l'ouverture du fichier !\n";
}
```

Ou bien, supprimer un fichier si celui-ci a une date de modification supérieure à 15 jours :

```
$file = '/home/gldavid/toto.txt';
OPEN FILE, "< $file" or die "Un problème est survenu pendant l'ouverture du fichier !\n";
if ( -M FILE > 15 ) {
  close FILE;
  unlink $file;
  print "$file supprimé\n";
}
```

Depuis la version 5.10 de perl, il est possible d'empiler les tests sur un fichier. Voici un exemple

```
my $fichier = "/home/djibril/fichier.txt";
if ( -e -f $fichier ) { print "Le fichier $fichier existe"; }
```

Le code ci-dessus vérifie que $fichier existe et soit un fichier.

Toutefois, les informations sur les fichiers peuvent être aussi accessibles à l'aide des fonctions stat et lstat. Pour simplifier, stat ne concerne que les fichiers, lstat vous renseignera plus en détail pour des liens symboliques. La fonction stat vous renvoie au total treize informations que l'on peut rassembler soit à l'aide de treize variables scalaires ou d'un tableau :

```
my $chemin = '/home/gldavid/toto.txt';
my
 ($dev, $ino, $mode, $nlink, $uid, $gid, $rdev, $size, $atime, $mtime, $ctime, $blksize, $block) = stat $chemin
# L'équivalent serait :
# @infos = stat $chemin;
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Ainsi, ces informations sont :

$dev ou $infos[0]	**Le numéro de périphérique, de "device" du système de fichiers**
$ino ou $infos[1]	**Numéro d'inode du fichier**
$mode ou $infos[2]	**Ensemble des bits de permission du fichier (ex : 0755). Droits du fichier (type et permissions)**
$nlink ou $infos[3]	**Nombre de liens durs du fichier**
$uid ou $infos[4]	**Identification numérique de l'utilisateur propriétaire du fichier**
$gid ou $infos[5]	**Identification numérique du groupe propriétaire du fichier**
$rdev ou $infos[6]	**L'identificateur du "device" (fichiers spéciaux uniquement)**
$size ou $infos[7]	**Taille totale du fichier en octets**
$atime ou $infos[8]	**Date (en secondes) d'accès au fichier depuis l'origine des temps**
$mtime ou $infos[9]	**Date (en secondes) de dernière modification du fichier depuis l'origine des temps**
$ctime ou $infos[10]	**Date (en secondes) de dernière modification de l'inode (pas la date de création) du fichier depuis l'origine des temps**
$blksize ou $infos[11]	**Taille de blocs préférée pour les E/S sur les fichiers, indication pour le système**
$block ou $infos[12]	**Nombre de blocs réellement occupés du fichier**

Petit rappel : lstat convient mieux pour les liens symboliques et renverra les mêmes informations que stat. Toutefois, si vous utilisez lstat sur un fichier, lstat renverra dans tous les cas les mêmes informations que stat.

- **Les inodes (contraction de "index" et "node"; en français : n?ud d'index) sont des structures de données contenant des informations concernant les fichiers stockés dans certains systèmes de fichiers, ce sont le centre de tous les échanges entre le disque et la mémoire ;**
- **set Group ID ;**
- **Set User ID ;**
- **Le sticky bit donne le droit de manier de façon plus subtile les droits d'écriture d'un fichier. En effet, un droit d'écriture signifie que l'on peut modifier et supprimer le fichier. Le sticky bit permet de faire la différence entre les deux.**

Se déplacer dans l'arborescence

Auteurs : GLDavid ,

Se déplacer dans une arborescence de répertoires sous Perl est tout aussi simple que dans un système Unix. Bien entendu, vous pourriez avoir recours à un appel système mais Perl vous fournit la commande vous permettant de vous déplacer où bon vous semble

```
$chemin = '/home/gldavid/documents';
chdir $chemin;
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
chdir '/home/2Eurocents/ultra_secrets';
```

Handle de répertoire

Auteurs : GLDavid ,

Ouvrir un répertoire n'est pas plus difficile que d'ouvrir un fichier. Encore une fois, nous aurons recours au handle, ces précieux médias de communication entre votre programme Perl et vos fichiers/répertoires. Mais, tout comme les handles de fichiers, n'oubliez jamais de fermer vos handles de répertoires :

```
my ( $nbrep, $nbfic, $fichier, $dossier ) = ( 0, 0, undef, '/home/gldavid/perl' );
opendir my $dir, $dossier or die "$dossier n'existe pas !";
my @files = readdir $dir;
foreach $fichier (@files) {
  if ( -f "$dossier/$fichier" ) { $nbfic++; }
  if ( -d "$dossier/$fichier" ) { $nbrep++; }
}
closedir $dir;
print "Il y a $nbfic fichier(s) et $nbrep répertoire(s) dans $dossier.\n";
```

Ce programme permet de compter le nombre de fichiers et de répertoires contenus dans le répertoire /home/gldavid/perl.
Avec l'instruction opendir, nous ouvrons notre handle DIR sur ce répertoire. Si celui-ci n'existe pas, le programme se termine avec un message d'erreur décrit dans la clause die. Si tout se déroule bien, la fonction readdir va rassembler dans le tableau files tous les fichiers/répertoires contenus dans notre répertoire d'étude. Remarquez que readdir accepte le nom de votre handle. Dans la boucle foreach, nous testons pour chacun des fichiers si celui-ci est un fichier (auquel cas, la variable $nbfic est incrémentée) ou un répertoire (la variable $nbrep est incrémentée). A la fin de cette boucle, nous affichons le nombre de fichiers et de répertoires contenus dans /home/gldavid/perl.

Suppression de fichiers

Auteurs : GLDavid , Djibril ,

Perl vous donne une fonction qui vous permettra d'effacer un fichier, pourvu que vous en soyez propriétaire :

```
$fichier = "/home/gldavid/toto.txt";
unlink $fichier;
unlink '/home/gldavid/secrets_gnux/faq.html';
```

La fonction unlink vous permet d'effacer un ou plusieurs fichiers car unlink prend en argument une liste de fichier.

```
unlink $fichiers;
unlink $fichier1, $fichier2, $fichier3;
unlink glob '*.mp3';
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

L'utilisation de la clause glob avec le motif "*.mp3" fera que la fonction unlink effacera tous les fichiers portant l'extension .mp3 du répertoire /home/gldavid/MP3.

Renommer des fichiers

Auteurs : GLDavid ,

Etes-vous anglophone ? Dans le domaine de l'informatique, cela peut vous être utile. En effet, pour renommer des fichiers, vous aurez recours à la fonction... rename !

```
$oldfile = '/home/gldavid/toto.txt';
$newfile = '/home/gldavid/tata.txt';
rename $oldfile, $newfile;
rename '/home/gldavid/tata.txt', '/home/gldavid/titi.txt';
```

lien : Source Comment renommer ou copier un fichier ?

Création/Suppression de répertoires

Auteurs : GLDavid , Djibril ,

Les unixiens ne sont vraiment pas dépaysés avec Perl ! La création de répertoires nécessite l'appel de la fonction mkdir. Pour supprimer un répertoire, vous aurez naturellement recours à la fonction rmdir si votre répertoire cible est vide de tout fichier :

```
mkdir '/home/gldavid/dossier';
rmdir '/home/gldavid/dossier';
$dossier = '/home/gldavid/perl';
mkdir $dossier;
rmdir $dossier;
```

lien : Source Comment copier ou supprimer un répertoire en perl ?

Des commandes Unix dans Perl pour les fichiers

Auteurs : GLDavid ,

Au chapitre des commandes Unix se retrouvant dans Perl pour la gestion des fichiers, on retrouve notamment la commande chmod qui permet de modifier les permissions de lecture, écriture, exécution d'un fichier.

```
chmod 0755, "/home/gldavid/toto.txt";
```

Dans cet exemple, nous attribuons au fichier /home/david/toto.txt toutes les permissions pour l'utilisateur et les permissions de lecture/exécution pour le groupe et les autres. Attention, les permissions symboliques du type r+w ne sont pas permises. Un petit rappel de bon aloi :

- **la lecture a un poids de 4 ;**
- **l'exécution a un poids de 1 ;**
- **l'écriture un poids de 2.**

De même, il est possible d'utiliser la fonction chown pour modifier le propriétaire du fichier :

```
chown "GLDavid", "root", /home/gnux/faq.html;
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Cette commande Perl indique que le fichier /home/gnux/faq.html a comme nouveau propriétaire GLDavid du groupe root.

Ouvrir et fermer un fichier (depuis Perl 5.6)

Auteurs : Djibril ,

Depuis Perl 5.8, il est possible et recommandé d'ouvrir les fichiers d'une façon beaucoup plus propre et moins sujette à des erreurs :

1. **utilisez des handles lexicaux (ou indirect) ;**
2. **utilisez la syntaxe open à trois arguments ou bien le module IO::File.**

Code propre version Perl > 5.6

```
my $fichier  = './MonFichier.txt';
my $fichier2 = './MonFichier2.txt';

# Lecture du fichier
open my $FH, '<', $fichier or die "Impossible de lire $fichier";

#...
#...
close $FH or die "Impossible de fermer $fichier";

# Ecrire dans un fichier
open my $FH2, '>', $fichier2 or die "Impossible d'écrire dans $fichier2";
print {$FH2} "Mon premier test\n";
close $FH2 or die "Impossible de lire $fichier2";
```

Autre méthode :

Utilisation du module IO::File

```
use IO::File;

my $fichier  = "./MonFichier.txt";
my $fichier2 = "./MonFichier2.txt";

# Lecture du fichier
my $FH = new IO::File $fichier, '<' or die "Impossible de lire $fichier";

#...
#...
close $FH or die "Impossible d'écrire dans $fichier";

# Ecrire dans un fichier
my $FH2 = new IO::File $fichier, '>' or die "Impossible d'écrire dans $fichier2";
print {$FH2} "Mon premier test\n";
close $FH2 or die "Impossible d'écrire dans $fichier2";
```

lien : IO::File

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > Perl avancé > Les processus

La fonction system

Auteurs : GLDavid , Djibril ,

Un processus est un appel au système pour exécuter un programme. Dans ce chapitre, nous allons envisager trois façons d'exécuter des processus. Dans un premier temps, intéressons-nous à la fonction system.
L'appel de cette fonction demande à Perl de lancer un processus enfant pour exécuter le programme appelé par system. Une fois que ce programme se termine, le processus enfant s'achève et rend la main au processus père à savoir votre programme :

```
system "ls";
```

Dans ce cas, nous appelons la commande Unix "ls" qui affichera son résultat sur la sortie standard.

```
my $commande = 'ls';
system($commande) == 0 or die "Erreur de la commande : $commande\n";
```

La fonction exec

Auteurs : GLDavid ,

La fonction exec est semblable à system. Mais, sa différence avec la fonction précédente réside dans le fait que le processus père se termine en même temps que le processus fils.

```
exec ("ls");
print "Et maintenant, qu'allons-nous faire ?";
```

Dans ce programme, Perl lance un processus fils qui exécutera la commande ls. Mais, contrairement à system, lorsque la commande ls se termine, le processus fils meurt ainsi que le processus père. exec est intéressant lorsque vous avez besoin de lancer via Perl des programmes particulièrement longs et pour lesquels vous aurez besoin de libérer des ressources rapidement.

Utiliser les backquotes

Auteurs : GLDavid ,

Utiliser les backquotes (ou apostrophes inversées pour les francophones) vous permettra de récupérer une sortie pour vos programmes. Par exemple :

```
$sortie = `ls`;
print $sortie;
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Ce petit programme permet à Perl de lancer un processus fils qui exécutera la commande ls. A la fin de ce programme, le processus fils meurt en rendant au processus père la sortie de la commande ls contenue dans la variable scalaire $sortie. Bien entendu, rien ne vous empêche d'utiliser des tableaux pour récupérer les sorties de vos programmes.

Fork you !

Auteurs : GLDavid ,

Amis du C, réjouissez-vous ! Votre fonction pour les threads favoris, à savoir fork, est aussi présente dans Perl. Pour les non-initiés, à quoi sert la fonction fork ?
Cette fonction permet de créer un processus enfant et de mettre le processus parent (votre programme Perl) en veille en attendant la fin de ce processus enfant. Elle duplique (clone) un processus en créeant une copie conforme de lui même :

```
defined($pid = fork) or die "Pas de fork possible : $!";
unless($pid) {
print "Processus fils.\n";
exec "ls";
}
waitpid($pid, 0);
print "Retour au père.";
```

Analysons ce code.
A la première ligne, la variable scalaire reçoit le résultat de fork. Si cette variable est nulle (undef pour Perl), une erreur est levée. Dans le cas contraire, $pid est défini, vaut zéro et le processus enfant est créé.
Tant que cette variable $pid demeure définie, le processus enfant reste en vie et exécute la tâche qui lui a été assignée : écrire une chaîne de caractères et lancer la commande système ls. Si vous avez bien suivi ce chapitre, vous aurez noté que cet appel utilise la fonction exec. Une fois que cette fonction termine sa tâche, le processus qui l'appelle se termine en même temps que la fonction. Ainsi, le processus enfant meurt (l'informatique est cruelle !) et retourne son code au numéro pid du père. D'ailleurs, que fait le processus père pendant l'exécution du fils ? Avec la fonction waitpid, celui-ci attend la valeur de retour pid de son fils. Une fois cette valeur obtenue, le processus père reprend la main et écrit une chaîne de caractères.

Les signaux

Auteurs : GLDavid ,

Un signal système est un signal envoyé à un processus. Par exemple, sous Unix, citons :

SIGHUP	**1**	**interruption de l'entrée**
SIGINT	**2**	**interruption clavier forte (Ctrl-C)**
SIGQUIT	**3**	**interruption clavier faible (Ctrl-\)**
SIGILL	**4**	**instruction illégale**
SIGABRT	**6**	**arrêt catastrophe d'un processus (abort())**
SIGKILL	**9**	**arrêt impératif**
SIGTERM	**15**	**terminaison normale d'un processus**

Perl vous permet ainsi de manipuler ces différents signaux et de les faire interagir avec vos processus.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.
http://perl.developpez.com/faq/perl/

Sommaire > Perl avancé > Les modules

Qu'est-ce qu'un module ?

Auteurs : Djibril ,

Les modules sont des librairies ou bibliothèques (code Perl) qui ajoutent des possibilités, des fonctionnalités au langage Perl. Ils sont stockés dans un fichier NOM.pm (pm pour Perl Module) et intégrés dans le programme par use NOM; (ou par require NOM;).

Appeler un module local

Auteurs : Djibril , GLDavid ,

Il peut être très utile pour vos codes de développer vos propres modules. Encore faut-il les appeler. En réalité, l'appel à un module local n'est pas si différent de l'appel d'un module que vous aurez installé sur votre machine. Considérons ce petit module :

essai.pm
```
#!/usr/bin/perl
package essai;
print "Bonjour tout le monde !\n";
1;
```

Dans le même répertoire que le module, vous écrivez votre script appelant ce module :

```
#!/usr/bin/perl

use warnings;
use strict;
print "Au revoir !\n";
use essai;
```

Dans cet exemple, vous afficherez d'abord la chaîne "Bonjour tout le monde !" puis celle de votre code. Mais il est possible aussi d'utiliser le mot-clef require pour appeler un module :

```
#!/usr/bin/perl
use warnings;
use strict;
print "Au revoir !\n";
require essai;
```

Cette fois-ci, c'est la chaîne de votre code qui sera affichée en première. La différence entre use et require se situe au niveau des temps d'appel. use sera appelé dès l'exécution de votre script Perl alors que require sera appelé conformément à la suite d'instruction de votre code. require se comporte ainsi comme une macro C *#include*. Voici un deuxième exemple. Le module "mon_module" contient dans notre cas un sous-programme (longueur) qui sera appelé dans le programme principal. Il aurait pu contenir 0 ou plusieurs sous-programmes. *Rappel : un module est une bibliothèque de sous-programmes ou procédures ajoutant des fonctionnalités à notre programme principal*.

mon_module.pm
```
#!/usr/bin/perl
use warnings;
use strict;

sub longueur {
  my ($seq) = @_; # récupération de la séquence à traiter
  my $lg = length $seq;
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

mon_module.pm
```
    print "c'est une séquence de longueur $lg\n";
}
1;  #obligatoire dans les modules en fin de programme.
```

script_principal.pl
```
#!/usr/bin/perl

use warnings;
use strict;
use lib './repertoire_module';  #mon module se trouve dans le répertoire "repertoire_module"
use mon_module; #=>utilisation du module "mon_module"

my $seq = "ATCGTGC";

#utilise la fonction &longueur de mon module
&longueur($seq);
print "c'est ok !";
```

Appeler un module annexe

Auteurs : GLDavid ,

De la même manière que nous avons appelé un module local, nous utiliserons encore le mot-clef use pour appeler un module externe. Prenons l'exemple du module File::Basename. Ce module va nous permettre de déterminer le non de base d'un fichier :

```
#!/usr/bin/perl
use strict;
use warnings;

use File::Basename;

my $file     = '/home/gldavid/Djibril.secrets';
my $basename = basename $file;
print "$basename\n";
```

Dans ce code, nous appelons une fonction basename contenu dans le module. Cette fonction prend en paramètre une chaîne de caractères relative à un nom de fichier. basename rendra ainsi le nom de base du fichier débarrassé de sa localisation soit, dans notre cas, Djibril.secrets.

lien : Source **Comment récupérer le nom (ou chemin) ou l'extension d'un fichier ?**

Listing de quelques modules utiles

Auteurs : Djibril ,

Voici une liste de modules Perl couramment utilisés pour améliorer et étendre les fonctionnalités de Perl.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Nom du module	Commentaires
CGI	Permet la création de page Web via la récupération des informations d'un formulaire
DBI	Accès aux bases de données (installez également le driver adéquat pour son fonctionnement)
DBD::Pg	Driver de bases de données Postgresql pour le module DBI
DBD::mysql	Driver de bases de données Mysql pour le module DBI
DBD::Oracle	Driver de bases de données Oracle pour le module DBI
XML::Twig	Permet de manipuler les fichiers XML
GD	Permet de créer des images GIF, JPG, PNG...
LWP::UserAgent	Utilitaires pour la consultation automatique de site web
HTML::Parser	Permet de manipuler les fichiers HTML
Net::FTP	Permet d'effectuer une connexion FTP
MIME::Lite	Permet d'envoyer des mails
Tk	Pour créer des interfaces graphiques

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > Perl avancé > Les modules > Installation des modules

Windows

Auteurs : Djibril ,

**Afin d'effectuer les installations des modules, vous devez utiliser le Gestionnaire de Package Perl (PPM). Ce dernier vous fournit une interface en ligne
pour gérer vos modules et extensions (packages) compatibles Win32. PPM vous permet d'accéder aux collections de packages, de les installer,
de les supprimer ou de les mettre à jour sur votre système.
Voici quelques explications sur l'installation, l'exécution et l'utilisation du Gestionnaire de Package Perl (PPM) :**

- **installation : PPM est installé en même temps que Perl pour Win32, donc rien à faire si Perl est bien installé bien sûr ;**
- **exécution : avant d'exécuter PPM, vous devez être connecté à Internet. En cas de souci, vérifier si votre connexion Internet utilise un firewall ou un proxy. Vous devriez positionner la variable d'environnement 'HTTP_proxy' avec le nom du serveur proxy.
  Ex : http_proxy=http://proxy.example.org
  Si votre serveur proxy nécessite un nom d'utilisateur et un mot de passe, les variables d'environnement "HTTP_proxy_user" et "HTTP_proxy_pass" devraient être initialisées avec ces valeurs ou bien positionnez "HTTP_proxy" ainsi :
  Ex: http_proxy=http://username:password@proxy.example.org
  Si de plus vous utilisez un port autre que le port 80, faites ainsi :
  http_proxy=http://username:password@proxy.example.org:8080
  PPM peut alors être exécuté en tapant "ppm" dans une fenêtre DOS ou bien, double clic sur C:\Perl\bin\ppm.bat et on accède à la même interface que sous Unix ;**
- **utilisation : par défaut, PPM utilisera la collection de package d'ActiveState, mais ceci peut être configuré en utilisant la commande "set" ou "rep" (cf. plus bas).**

**Une fois dans votre fenêtre console, vous aurez un prompt "ppm>".
L'installation s'effectue via la commande "install package_name".
Au fur et à mesure de l'évolution de Perl, l'utilisation de PPM s'est améliorée.
Je vais donc vous présenter son utilisation pour les versions de Perl inférieures
et supérieures à Perl 5.8.8 built 819.
Je vous conseille de lire les deux car même si le principe diffère, le but est le même.**
Pour les versions de Perl inférieures à 5.8.8 built 819
Exemple d'utilisation de PPM pour l'installation

```
ppm> install CGI
=============
Install 'CGI' version 2.91 in ActivePrl 5.8.2.808.
=============
Downloaded 147245 bytes.
Extracting 21/21: blib/arch/auto/CGI/.exists
Installing C:\Perl\html\site\lib\CGI.html
.....................................=> Installation OK
.....................................
Installing C:\Perl\site\lib\CGI.pm
.....................................
.....................................
Successfully installed CGI version 2.91 in ActivePerl5.8.2.808.
```

Bravo, l'installation du module CGI s'est bien déroulée. Essayons de le réinstaller par curiosité.

```
ppm> install CGI
Note: Package 'CGI' is already installed
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Pas besoin d'être brillant en anglais, on comprend bien que ce module est déjà installé !
Problèmes souvent rencontrés
Il arrive très souvent de se trouver face à un problème d'installation de modules avec plein de messages d'erreurs. Alors, on est un peu perdu, surtout quand on débute en Perl (ou dans l'installation de modules).
- Assurez-vous d'être connecté à Internet, car l'installation de modules via ppm par défaut nécessite que ce dernier aille chercher le module que vous souhaitez dans un dépôt ou encore repository en anglais *(la notion de repository sera expliquée plus bas)*.

- **Certaines versions de ppm peuvent être sensibles à la casse, donc faites attention.**
- **Il se peut que le module que vous cherchez à installer n'existe pas (faute de frappe par exemple).**

```
ppm> install toto
Error: Package 'toto' not found. Please 'search' for it first.
```

- Attention, ce sera ppm install DBD-mysql et non ppm install DBD::mysql
- Comme tous les perléens, vous avez l'habitude de vous rendre sur le site officiel CPAN pour faire une recherche de modules intéressants. Ensuite, tout content de les avoir trouvés, vous essayez de les installer et paf, la phrase magique "*ça marche pas*", avec pour messages sur le terminal :

```
ppm> install PDF::FromHTML
Error: Package 'PDF::FromHTML' not found. Please 'search' for it first.
ou bien
ppm> install PDF::FromHTML
ppm> searching for 'PDF::FromHTML' returned no results. Try a broder search first
ou
ppm> install GD
ppm> Error: PPD for 'GD.ppd' could not be found.
ou autre...
```

Alors, c'est là qu'intervient la notion de repository.
Les dépôts ou repository
En fait, lorsque vous installez un module en Perl via ppm, ce dernier va chercher le module en question dans le repository d'ActivePerl.
Ce dernier est ActiveState Package Repository. Il se peut que le module que vous avez vu sur le CPAN n'y soit pas présent.
Pour faire une recherche d'un module dans ses repositories, faites par exemple :

```
ppm> search DBD
Searching in Active Repositories
   1. Bundle-DBD-CSV          [0.1016] A bundle to install the DBD::CSV driver
   2. DBD-ADO                   [2.95] DBD-ADO
   3. DBD-ADO                   [2.95] A DBI driver for Microsoft ADO (Active Data Objects)
   4. DBD-ADO                   [2.94] ADO driver for the DBI module.
   5. DBD-AnyData               [0.08] DBD-AnyData
   6. DBD-AnyData               [0.08] DBI access to XML, CSV and other formats
   7. DBD-AnyData               [0.08] DBI access to XML, CSV and other formats
   8. DBD-Chart                 [0.80] DBD-Chart
   9. DBD-Chart                 [0.80] DBI driver abstraction for Rendering Charts and Graphs
  10. DBD-Chart                 [0.81] Chart driver for DBI module
  11. DBD-CSV                   [0.22] DBD-CSV
  12. DBD-CSV                   [0.22] DBI driver for CSV files
  13. DBD-CSV                   [0.21] DBI driver for CSV files
  14. DBD-Excel                 [0.06] DBD-Excel
  .....
```

Vous obtenez la liste des modules correspondant à votre recherche. Si vous ne trouvez pas ce que vous cherchez alors que vous l'avez vu sur le CPAN, il va falloir rajouter des repositories sur votre Windows.
Pour voir la liste de ces repositories :

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
ppm> rep (ou repository)
Repositories:
[1] ActiveState PPM2 Repository
[2] ActiveState Package Repository
```

Voici comment rajouter divers dépôts comme theoryx5 ou bribes qui répondront peut-être à vos besoins. Ainsi, durant l'installation, la recherche sera effectuée sur l'ensemble des dépôts que vous possédez.

```
ppm> rep add bribes http://www.bribes.org/perl/ppm
ppm> rep add threoryx http://theoryx5.uwinnipeg.ca/ppms/
```

Ainsi, votre recherche de modules se fera sur 4 et non 2 repositories.

```
ppm> repository
Repositories:
[1] ActiveState PPM2 Repository
[2] ActiveState Package Repository
[3] bribes
[4] threoryx

=> Il existe d'autres dépôts, à vous de chercher !!!
```

Une autre façon d'installer un module présent dans un repository consiste à taper *ppm install http://url/nom_package.ppd* si vous connaissez son adresse exacte bien sûr !! Exemple :

```
ppm> install http://www.bribes.org/perl/ppm/CGI.ppd
```

Le module CGI installé sera celui de bribes et non du CPAN. Mais ne vous inquiétez pas, ce sont des dépôts corrects et les modules sont généralement à jour.
CPAN n'est qu'une interface permettant de regrouper la plupart des modules à disposition, les documents et les différents miroirs vers ces modules.
Voilà, pour plus d'informations sur les ppm, voici un très bon site !
Après toutes ces indications, si vous avez toujours un souci avec les fichiers ppd, vous pouvez le créer vous-même, mais cela dépasse le domaine de cette FAQ :-) !!
Vous pouvez également télécharger le module à partir du CPAN et l'installer "à la UNIX" (voir plus bas), mais vous ne pourrez pas faire de make (inconnu de Windows) à moins de passer par cygwin ou Visual Studio.
Pour les personnes utilisant une version de Perl très ancienne (Perl 5.00), je vous conseille d'installer la version récente de Perl (actuellement 5.8.8). Cela vous évitera des soucis de version de ppm (tapez *ppm version* => 3.4 actuellement) et de compatibilité avec les modules récents.
Si d'autres messages d'erreurs persistent, soit il n'existe pas, soit cette FAQ ne vous sera pas suffisante et de ce fait un bon Google s'impose !
Pour les versions de Perl 5.8.8 built 819 et plus
Depuis la version 5.8.8, lorsque vous installez Perl depuis ActiveState, vous disposez de la version 4.01 de ppm

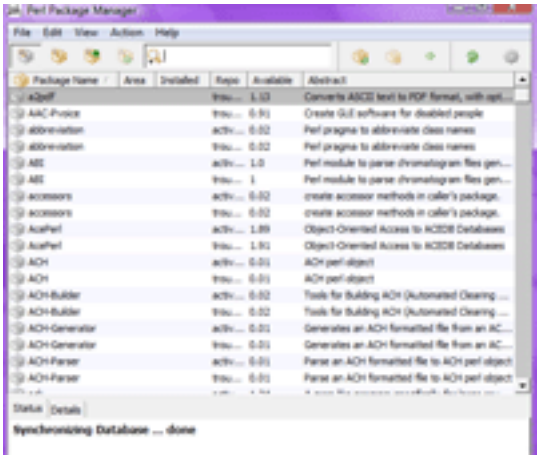
```
>ppm version
ppm 4.01
Copyright (C) 2007 ActiveSate Software Inc. All rights reserved.
```

L'installation des modules Perl a donc été simplifiée via une interface Tk. Pour toute installation de modules Perl, taper dans une fenêtre DOS la commande suivante :

```
>ppm
```

=> Une fenêtre Tk va s'ouvrir et va charger vos différents repositories.
Vous devez donc avoir bien évidemment Internet de disponible.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.



Interface ppm

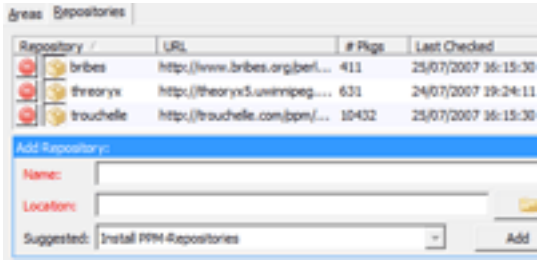
Dans un premier temps, je vais vous demander de rajouter certains repositories importants. Dans la fenêtre TK, cliquez sur le symbole le plus à droite tout en haut

symbole repository

Vous verrez que vous ne disposez que du repository *ActiveState Package Repository*
url : http://ppm4.activestate/MSWin32-x86/5.8/820/package.xml
Maintenant ajoutez les repositories suivants :

listing repository

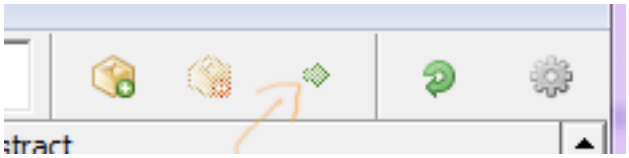
```
bribes     => http://www.bribes.org/perl/ppm/
  threoryx   => http://theoryx5.uwinnipeg.ca/ppms/
  trouchelle => http://trouchelle.com/ppm/
```



repository

Pour installer un nouveau module, revenez sur la fenêtre Tk principale et choisissez dans la liste des modules celui que vous souhaitez installer.
Si ce dernier est jaune (=> il est déjà installé), s'il est grisé alors il n'est pas installé.
Faites un clic droit et choisissez '*install*'. Vous pouvez le faire sur tous les modules que vous souhaitez installer.
Maintenant tous les modules que vous avez souhaité installer seront décorés d'une flèche verte. Cliquez maintenant sur la flèche verte

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.



fleche_verte

L'installation commence et gère toute seule les dépendances.
Voilà. Idem pour les désinstallations ou mises à jour des modules.

lien : Installation des modules Perl CPAN, Sous Windows, Linux et Mac OS
lien : Perl et modules CPAN

Unix

Auteurs : GLDavid , Djibril ,

- **Installation par téléchargement du module**

La meilleure façon d'installer un module sur une station Unix est encore de télécharger le module et de le compiler.
Dans cet exemple, nous allons installer un module que nous verrons aussi plus tard, le module CGI qui permet le traitement des formulaires HTML.
Pour le moment, il faut télécharger ledit module (au format tar.gz) puis le décompresser. Dans une console shell.

```
wget http://search.cpan.org/CPAN/authors/id/L/LD/LDS/CGI.pm-3.07.tar.gz
tar -xvzf CGI.pm-3.07.tar.gz
cd CGI.pm-3.07
```

Dans le répertoire du module se trouvera toujours un fichier README sur la façon d'installer le module.
Lisez-le ! Pour les non-anglophones, voici la procédure la plus couramment utilisée :

```
#Vérification de votre environnement et de votre Perl et écriture du Makefile
perl Makefile.PL
#Compilation
make
#Installation en étant ROOT !
make install
#Have fun !
```

- **Installation via CPAN**

Le défaut majeur de l'installation précédente est la gestion des dépendances. Il arrive très souvent que le module que vous souhaitez installer dépende d'autres modules et dans ce cas, il vous sera demandé d'installer un autre et ainsi de suite. Ce qui bien sûr peut devenir très lourd à gérer.
Je vous suggère de vous servir de l'utilitaire CPAN.
tapez la commande suivante :

CPAN
```
> CPAN
```

Il vous posera plusieurs questions et la plupart de temps, il faudra choisir la réponse par défaut.
Une fois CPAN configuré, pour installer un module, vous devrez taper :

```
> CPAN
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
cpan> install MON::MODULE
# faites un help pour voir toutes les commandes disponibles
cpan>help
```

Voilà !!!

lien : Installation des modules Perl CPAN, Sous Windows, Linux et Mac OS

lien : Perl et modules CPAN

Macintosh

Auteurs : Stoyak ,

- **Installation de Perl par défaut**

Le module CPAN.pm est un module préinstallé sous Mac OS X. Il a pour but d'automatiser le téléchargement et la construction de vos modules.
Pour monter un shell CPAN interactif, placez-vous dans votre home, et tapez l'instruction suivante :

```
perl -MCPAN -e shell
```

La toute première fois, une série de questions vous seront posées pour configurer le système. Dans la plupart des cas, les réponses par défaut suffisent. Elles seront sauvegardées dans *System/Library/Perl/CPAN.pm* ; Le premier module à télécharger est bundle libnet

```
install Bundle ::libnet
```

L'ensemble des modules installés par défaut ou par vos soins se trouve dans le répertoire */System/Library/Perl/*.

- **Version ActiveState**

Pour installer des modules avec ActivePerl, rien de plus simple ! Dans la fenêtre Terminal :

```
ppm install mon::module
```

Si vous n'êtes pas root sur votre machine, faites :

```
sudo -u  root  /usr/local/ActivePerl-5.8/bin/ppm3-bin  install  mon::module
```

- **Version Unix (troisième et dernière méthode !)**

Si les deux solutions précédentes n'ont pas été concluantes, si vous avez porté votre choix sur un module qui n'est pas disponible par le ppm d'ActiveState, si vous avez des problèmes avec la méthode par défaut... ne perdez pas espoir !! Il reste une dernière solution ! Souvenez-vous, votre Terminal est un Unix... Donc vous pouvez installer vos modules à la manière Unix !! Téléchargez votre module, décompressez-le dans n'importe quel de vos répertoires (votre home, pourquoi pas !). Ensuite, placez-vous dans le répertoire du module obtenu (cd mon_module) et faites :

```
perl Makefile.pl
make
sudo make install
```

Mais attention ! Il se peut que vous obteniez le message suivant :

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
-bash: make: command not found
```

Pas d'angoisse ! Make n'est pas reconnu par votre OS car vous n'avez pas installé les Xcode tools... Eh oui, ce n'est pas fait par défaut !!
Installez-les par un petit double-clic sur Xcode Tools.mpkg qui se trouve dans le répertoire Applications/Installers/Xcode Tools/Xcode Tools.mpkg et le souci est résolu !!
et voilà !!! votre module est installé !
Vous trouverez votre bonheur parmi ces trois variantes j'espère !!!
A vous de jouer !!

lien : Installation des modules Perl CPAN, Sous Windows, Linux et Mac OS
lien : Perl et modules CPAN

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Module DBI

Auteurs : Djibril ,

Le module DBI (database interface) permet de faire des connexions à une ou plusieurs bases de données telles Mysql, Postgresql, Oracle, Sybase... De ce fait, il vous sera possible d'intégrer des requêtes SQL dans vos scripts.
***Petit rappel : n'oubliez pas d'installer le module DBI et le driver*. Voici une liste de commandes à inclure dans le code Perl pour l'accès aux bases de données.**

```
use DBI;

# En début du programme, il permet de faire appel au module DBI et
# spécifie qu'on va faire un accès aux bases de données.
$db = DBI->connect( "dbi:mysql:toto", 'login', 'mot_de_passe' );

# Requêtes du genre "create...", "update...", "insert into..." => $db->do("requête");
$db->prépare("requête"); # préparation de la requête
$db->exécute(); # exécution de la requête
$db->fetchrow_array(); # ligne retournée par la requête (ou lignes dans une boucle)
$db->finish(); # fin de la requête
# Requêtes du genre "select..." Dans ce cas, 4 étapes doivent être exécutées :
# Connexion à une base de données mysql s'appelant "toto" et dont l'accès nécessite un
# login et mot de passe. Exécution d'une requête SQL. En fonction du type de requête,
# la syntaxe n'est pas la même. On différentie deux types d'écriture.
$db->disconnect; # pour se déconnecter de la base de données en fin de programme.
```

Exemple: Insertion d'une ligne dans une table sql et récupération de données.

Voici une table sql

```
titre        |acteur   |jour
--------------------------------
|shrek       |eddy     |lundi   |
|couleur     |hopkins  |mardi   |
|coup_foudre |robert   |mercredi|
--------------------------------
```

```
#!/usr/bin/perl
use strict;
use warnings;

use DBI;

#connexion à la base de données postgresql premieredb
my $dbp = DBI->connect( "dbi:Pg:dbname=premieredb", "login", "mot_de_passe" )
  or die "connexion impossible !";

#insertion d'une donnée
$dbp->do("insert into film values ('test_titre', 'test_acteur','test_jour');")
  || die "pb de requête : $DBI::errstr";

#requete postgresql
my $requete = "select acteur, jour from film;";

#prépare la requête sql
my $version = $dbp->prepare($requete);

#exécution de la requête sql
$version->execute() || die "pb de sélection : $DBI::errstr";

while ( my ( $acteur, $jour ) = $version->fetchrow_array ) {
  print "result: $acteur du jour $jour\n";
}
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
#spécifie la fin de la requête
$version->finish();

# déconnexion à la base de données
$dbp->disconnect();
```

résultat

```
result: eddy du jour lundi
result: hopkins du jour mardi
result: robert du jour mercredi
result: test_acteur du jour test_jour
nouvelle table
 titre        |acteur     |jour
-----------------------------------
|shrek        |eddy       |lundi    |
|couleur      |hopkins    |mardi    |
|coup_foudre  |robert     |mercredi |
|test_titre   |test_acteur|test_jour|
-----------------------------------
```

lien : Documentation officielle du module DBI

lien : Perl et les bases de données (DBI)

Module CGI

Auteurs : GLDavid , Djibril ,

Le module CGI (Common Gateway Interface) est particulièrement utile pour l'écriture de pages Web au format HTML mais surtout à la récupération et au traitement de données provenant de formulaires. Prenons un exemple simple. Soit la page HTML suivante notée accueil.html :

```
<HTML>
<BODY>
Rentrez votre prénom : <BR>
<FORM NAME="form" ACTION="/cgi-bin/script.pl"
METHOD="POST">
<INPUT TYPE="text" NAME="prenom">
<INPUT TYPE="submit" VALUE="Envoyer">
</FORM>
</BODY>
</HTML>
```

Dans cette page, on crée un formulaire contenant un champ de texte à compléter. Le deuxième élément est le bouton de soumission des données vers le script pointé par la clause ACTION du formulaire. Il s'agit de notre script Perl nommé script.pl et se situant dans le répertoire cgi-bin de notre serveur Web (notre page aurait pu aussi s'appeler *script.cgi* mais n'oubliez pas de modifier votre lien dans la page précédente) :

```
#!/usr/bin/perl
use strict;
use warnings;
use CGI;

my $cgi = new CGI;

# Création de l'entête de la page web
print $cgi->header();
print $cgi->start_html( -title => 'Titre de ma page', );

# Recupération du prénom
my $prenom = $cgi->param("prenom");
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
print "Bonjour, votre pr&eacute;nom est : <b>$prenom</b><br><br>\n\n";
print "Merci";

# Fermeture de la page web
print $cgi->end_html;
```

Exemples d'erreurs :
Il arrive souvent que vous ayez ce message : Internal Server Error.
Ceci indique qu'il y a une erreur dans votre script CGI. Le premier réflexe à avoir est d'aller regarder ses fichiers log. Le chemin vers le fichier log est indiqué dans le fichier de configuration de votre Apache. C'est le fichier httpd.conf, apache.conf ou apache2.conf... tout dépend du serveur Apache que vous avez installé.

Sous EasyPHP :
- **Fichier de configuration : C:\Program Files\EasyPHPX-X\conf_files\httpd.conf ;**
- **fichier log : C:\Program Files\EasyPHPX-X\apache\logs\error.log.**

Sous Linux :
- **Fichier de configuration : faites un find httpd.conf ou find apache.conf ou apache2.conf ;**
- **fichier log : c'est souvent /var/log/apache(2)/errors.log.**

A vous de chercher.
Lisez le fichier log, et essayez de comprendre le message d'erreur Perl.
Pensez à vérifier que l'entête HTML est bien créé dans votre script CGI, que les fins de lignes de vos programmes sont bien des retours chariots Linux et non Windows (Source Compatibilité Unix/Mac/Linux/Windows des fichiers (^M)).

Les erreurs les plus fréquentes que l'on peut trouver lors d'une demande d'ouverture de page web :
- **erreur 400 : mauvaise requête (Bad request) ;**
- **erreur 401 : autorisation refusée (Authorization) ;**
- **erreur 403 : répertoire interdit (Forbidden) ;**
- **erreur 404 : fichier non trouvé (File not found) ;**
- **erreur 500 : erreur de configuration (Internal server error).**

lien : Source Compatibilité Unix/Mac/Linux/Windows des fichiers (^M)
lien : Documentation en français sur l'utilisation du module CGI

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Windows

Auteurs : Djibril ,

- **Pour les versions de Perl inférieures à 5.8.8 built 819**

Comme pour l'installation des modules Perl, vous devez taper la commande suivante :

```
ppm
```

une fois le prompt ppm> suivant affiché, vous devrez taper de multiples commandes.
Vous trouverez toutes les commandes nécessaires sur le site d'ActiveState (bribe ppm ou ppm ActiveState).
Voilà ce que je fais personnellement :
- Pour voir si un module est à jour (Exemple de DBI)

```
ppm>upgrade DBI
et j'obtiens ceci :
DBI 1.48: up to date.
Donc ce module est à jour.
```

- Pour voir si tous les modules installés sur ma machine sont à jour

```
ppm>upgrade
ou
ppm>upgrade > mise_a_jour.txt
```

Vous obtiendrez la liste de tous vos modules (que je préfère mettre dans un fichier texte grâce à la redirection. Il est bien évident que le fichier mise_a_jour.txt sera créé dans le répertoire où vous avez tapé ppm. Voici un exemple de résultat :

```
Spreadsheet-WriteExcel 2.11: new version 2.14 available in ActiveState PPM2 Repository
Test-Simple 0.47: up to date.
Tie-Gzip 0.06: up to date.
Tk 800.024: new version 804.026 available in theoryx
UNIVERSAL-exports 0.03: up to date.
XML-Parser 2.34: up to date.
XML-Simple 2.09: new version 2.14 available in theoryx
```

Vous remarquerez que les modules *Test-Simple, Tie-Gzip, UNIVERSAL-exports et XML-Parser* sont à jour. Il existe par contre une nouvelle version du module *Spreadsheet-WriteExcel* dans ActiveState PPM2 Repository et, pour *Tk et XML-Simple*, une nouvelle version est disponible dans theoryx (Repository que j'avais créé avant l'installation de mes modules, cf. : installation module sous Windows)
- Mettre à jour mes modules installés sur ma machine
Comme précédemment un ou tous mes modules :

```
ppm>upgrade -install  -precious Archive-Tar  => mise à jour du module Archive-Tar
ou
ppm>upgrade -install => mise à jour de tous les modules
```

Je tiens encore à préciser que tous les modules seront mis à jour en fonction de ceux présents dans vos dépôts (c'est-à-dire ActiveState par défaut et vos rajouts).
Amusez-vous à créer un script Perl pour gérer ces mises à jour, c'est un bon exercice.
Voilà, bon courage !!

- **Pour les versions de Perl 5.8.8 built 819 et plus**

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Comme expliqué dans la section Windows, tout se fait via l'interface Tk de façon beaucoup plus simple.

lien : bribe ppm
lien : ppm ActiveState

Linux

Auteurs : GLDavid ,

La mise à jour de vos modules peut être réalisée en compilant la dernière version de votre module téléchargé depuis le CPAN.
Autrement, vous pouvez aussi invoquer la commande suivante pour vous assister dans vos mises à jour :

```
perl -MCPAN
```

Macintosh

Auteurs : Djibril ,

Si vous utilisez ActivePerl, référez-vous aux mises à jour sous Windows ci-dessus !
Si non vous mettez à jour vos modules comme sous Unix/Linux (en recompilant le nouveau module téléchargé) ou via les commandes MCPAN ; faute d'expérience, tapez :

```
perl -MCPAN
help
```

Vous aurez des informations !!
Bon courage !

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Modules CPAN indispensables

Auteurs : Djibril , 50Nio ,

Voici une liste des modules CPAN indispensables que vous devez toujours avoir sous la main :

- **Data::Dumper : permet d'afficher n'importe quelle variable, très utile pour deboguer un programme ;**
- **File::Basename : pour récupérer les noms des fichiers et les chemins ;**
- **File::Copy : pour faire une copie des fichiers ;**
- **File::Copy::Recursive : pour faire une copie des fichiers et répertoires (récursivement) ;**
- **File::Temp : pour créer des fichiers temporaires proprement ;**
- **Log::Log4perl : gestion des logs ultrasimple et pratique. A utiliser dès que vos daemons atteignent une certaine complexité ;**
- **Net::FTP : pour faire du FTP ;**
- **Time::Local : gestion du time (Epoch et autres) ;**
- **Date::Calc et Date::Time : manipulation des dates, comparaison des dates et autres ;**
- **DBI, DBD::Mysql, DBD::Pg, DBD::Oracle... : gestion des bases de données de façon simple et efficace ;**
- **CGI : gestion et création d'interface web ;**
- **et bien d'autres à chercher sur le site du CPAN.**

Ayez le réflexe CPAN !!

lien : Recherche CPAN

Comment savoir si un module est disponible dans le Core de Perl ?

Auteurs : Philou67430 ,

Il suffit d'appeler corelist sous une console.

exemple avec le module POSIX

```
corelist POSIX
```

résultat

```
POSIX was first released with Perl 5
```

N.B. Si vous disposez d'une version de Perl inférieure à 5.8.9, c'est-à-dire Perl 5.8, 5.6, ou autres, vous devrez installer le module Module::CoreList afin de pouvoir jouer avec cet utilitaire.

lien : Module::CoreList

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > Perl avancé > Les références

Qu'est-ce qu'une référence, à quoi ça sert ?

Auteurs : Djibril ,

Les références Perl sont des variables qui permettent de référencer d'autres variables, tableaux, hashs, fonctions ou handles. La notion de référence peut être vue comme les pointeurs en C. Mais pas de panique :-(, c'est beaucoup plus simple que ce que vous croyez ! Vous allez me dire, oui, c'est bien beau, mais à quoi ça sert ? Voici donc un exemple...

```
my @tab1 = (1, 2, 3);
my @tab2 = ('a', 'b', 'c');

affiche(@tab1, @tab2);

sub affiche {
  my ( @recup1, @recup2 ) = @_;
  print "Voici votre tableau tab1 récupéré dans ma fonction : @recup1\n";
  print "Voici votre tableau tab2 récupéré dans ma fonction : @recup2\n";

  return;
}
```

Résultat :

```
Voici votre tableau tab1 récupére dans ma fonction : 1 2 3 a b c
Voici votre tableau tab2 récupére dans ma fonction :
```

Bizarre le résultat non ? Dans notre exemple, deux tableaux @tab1 et @tab2 sont donnés en argument à notre fonction qui a pour unique but de les réafficher. Vous remarquez qu'il n'arrive pas à distinguer vos tableaux et à les récupérer correctement.
Il en aurait été de même avec des hashs.
Petite explication :
@_ récupère une liste des données (@tab1 et @tab2 concaténés). De ce fait, toutes les données récupérées se retrouvent dans @recup1, @recup2 reste vide. Comment doit-on faire pour que @recup1 et @recup2 soient corrects ? Facile ! Vous avez trouvé ? Les références !!! On passera en argument à la fonction les références de nos tableaux.

Comment créer une référence ?

Auteurs : Djibril ,

Nous avons bien compris avec l'exemple ci-dessus l'utilité des références. La question est maintenant de savoir comment faire pour désigner (créer une référence) un tableau, un hash, une fonction...
Rien de plus simple, il suffit juste de placer un backslash ("\") devant la variable à référencer. Ainsi, la référence $ref_tab de notre tableau @tab s'écrit

```
$ref_tab = \@tab;
```

Exemples :

```
my $ref_tableau  = \@mon_tableau;
my $ref_hash     = \%mon_hash;
my $ref_fonction = \&ma_fonction;
my $ref_handle   = \*FILE;
```

Voilà, c'est aussi simple que ça. Par curiosité, essayez cet exemple :

```
my @mon_tableau = ( 1, 2, 3 );
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
my $ref_tab = \@mon_tableau;
print $ref_tab;
```

Vous obtenez ce genre de résultat : ARRAY(0x225da0).
C'est l'adresse, la référence du tableau @mon_tableau. D'ailleurs, aussi doué que vous soyez, vous avez déjà dû rencontrer des erreurs avec ce genre de caractère !! Généralement, sans le savoir, c'est dû au fait qu'on pense travailler sur un tableau ou une valeur de ce dernier alors que l'on travaille sur sa référence. Sinon, il existe une autre façon de créer des références, qu'on appelle références anonymes, voir ci-dessous.

Qu'est-ce qu'une référence anonyme ?

Auteurs : Djibril ,

Excusez mon vocabulaire, mais c'est une façon un peu tordue de créer une référence ! En fait, on crée une référence sur une variable n'existant pas au préalable, ne possédant pas de nom. Je sais, ce n'est pas clair... Ce sont des références sur des structures anonymes en vue d'obtenir des structures de données plus complexes. Un petit exemple vous éclaira, du moins je l'espère !

```
my $ref_tab = [ 1, 2, 3 ];
my $ref_hash = {
  "cle1" => "valeur1",
  "cle2" => "valeur2",
};
print $ref_tab;
print $ref_hash;
```

Il vous affichera deux adresses. Eh oui, c'est tordu comme raisonnement ! La première pour le tableau (1,2,3) sans nom, la deuxième pour le hash ("cle1" => "valeur1", "cle2" => "valeur2") sans nom.
Vous remarquez les couleurs rouges pour que vous puissiez prêter attention à la façon d'écrire les références anonymes.
Petit résumé : un tableau s'écrit (1,2,3), une référence anonyme à un tableau [1,2,3].
Bon, c'est bien tout ce blabla, mais comment les utiliser ? On y arrive, on y arrive, allez voir ci-dessous (déférencement).

Comment utiliser les références, le déférencement !

Auteurs : Djibril , Stoyak ,

Ah enfin !!! Après toutes ces explications, vous allez enfin savoir utiliser ces fameuses références !! Tout d'abord, sachez qu'il existe deux façons de les utiliser (eh oui, c'est Perl, avec sa grande capacité à pouvoir écrire du code de différentes façons) :
1) Première façon
@{$ref_tableau} est identique à @tableau.
${$ref_tableau}[0] est identique à $tableau[0]
%{$ref_hash} est identique à %hash
${$ref_hash}{"cle1"} est identique à $hash{"cle1"}
Voici encore plus simple si vous voulez, mais plus bizarre quand on tombe dessus pour la première fois.
Vous pouvez omettre les accolades si $ref a été déclaré : my $ref = \@tableau;
@{$ref_tableau} est identique à @$ref_tableau.
${$ref_tableau}[0] est identique à $$ref_tableau[0]
%{$ref_hash} est identique à %$ref_hash
${$ref_hash}{"cle1"} est identique à $$ref_hash{"cle1"}
Résumé:
@{$ref_tableau} est identique à @$ref_tableau, lui-même identique à @tableau ! En d'autres termes, un tableau est identique au tableau de sa référence ! C'est clair, non ?

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

- **$ref=\@tab => création d'une référence, c'est le référencement**
- **@{$ref} => c'est le déférencement.**

Exemple :

```
# Déclaration de mon tableau
my @mon_tableau = ( 1, 2, 3 );

# Déclaration de la référence de mon tableau
my $ref_tab = \@mon_tableau;

print @{$ref_tab};    #ou print @$ref_tab;

#  et vous obtiendrez : 123
```

En résumé, il faut juste mettre {$ref} à l'endroit où on met d'habitude le nom du tableau ou d'un hash.
2) Deuxième façon
C'est une écriture simplifiée (ressemblant à l'écriture objet pour les connaisseurs). Cette méthode s'utilise surtout lorsque l'on travaille sur un élément de tableau et de hash.
Exemple :

```
my @tableau = ( 1, 2, 3 );
my $ref_tableau = \@tableau;

my %hash = ( "cle1" => "valeur1", "cle2" => "valeur2" );
my $ref_hash = \%hash;
```

$ref_tableau->[0] est identique à ${$ref_tableau}[0] identique à $tableau[0] égal à la valeur 1
$ref_hash->{"cle1"} est identique à ${$ref_hash}{"cle1"} identique à $hash{"cle1"} égal à la valeur valeur1
Voilà, est-ce compliqué ??
Va falloir relire la doc plusieurs fois au début !! C'est normal !!
Bon pour la route, reprenons notre tout premier exemple (dans la section Qu'est-ce qu'une référence, à quoi ça sert ?) :

```
my @tab1 = ( 1,   2,   3 );
my @tab2 = ( 'a', 'b', 'c' );

# On donne en argument à la fonction affiche les références à nos tableaux.
affiche( \@tab1, \@tab2 );

sub affiche {

  # On récupère les références
  my ( $ref1, $ref2 ) = @_;

  # Affichage
  print "Voici votre tableau tab1 récupéré dans ma fonction : @{$ref1}\n";
  print "Voici votre tableau tab2 récupéré dans ma fonction : @{$ref2}\n";
}
```

```
résultat :

Voici votre tableau tab1 récupéré dans ma fonction : 1 2 3
Voici votre tableau tab2 récupéré dans ma fonction : a b c
Magique, non !!
```

Attention : sachez tout de même qu'après l'appel de votre fonction, si celle-ci doit traiter et modifier le tableau, le tableau d'origine sera modifié !
Exemple :

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
my @tab1 = ( 1,   2,   3 );
my @tab2 = ( 'a', 'b', 'c' );

# On donne en argument à la fonction affiche les références à nos tableaux.
affiche( \@tab1, \@tab2 );

print "Voici votre tableau tab1 récupéré dans ma fonction : @tab1\n";
print "Voici votre tableau tab2 récupéré dans ma fonction : @tab2\n";

sub affiche {

  # On récupère les références
  my ( $ref1, $ref2 ) = @_;

  push @{$ref1}, "dudu";
  push @{$ref2}, "dudu";
}
```

```
résultat :

Voici votre tableau tab1 récupéré dans ma fonction : 1 2 3 dudu
Voici votre tableau tab2 récupéré dans ma fonction : a b c dudu
```

Si vous ne souhaitez pas modifier les tableaux d'origine, à vous de créer vos tableaux dans vos fonctions :

```
.............
.............
print "Voici votre tableau tab1 récupéré dans ma fonction : @tab1\n";
print "Voici votre tableau tab2 récupéré dans ma fonction : @tab2\n";

sub affiche {

  # On récupère les références
  my ( $ref1, $ref2 ) = @_;

  # Copie de mes tableaux pour que la modification ne soit appliquée que dans la procédure.
  my @tableau1_interne = @{$ref1};
  my @tableau2_interne = @{$ref2};
  push @tableau1_interne, "dudu";
  push @tableau2_interne, "dudu";
}
```

```
résultat :

Voici votre tableau tab1 récupéré dans ma fonction : 1 2 3
Voici votre tableau tab2 récupéré dans ma fonction : a b c
```

Voilà !!! Les tableaux @tab1 et @tab2 n'ont pas été modifiés !

Comment parcourir un hachage contenant des références ?

Auteurs : Djibril ,

Prenons un exemple de hachage contenant des références, nous allons le parcourir.

```
#!/usr/bin/perl
use strict;
use Carp;
use warnings;

my %HashReferenceComplex = (
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
    'niv1' => 'valeur 1',
    'niv2' => [ 'case0', 'case1', 'case2', 'case3' ],
    'niv3' => {
      'niv3-1_a' => 'valeur3-1',
      'niv3-1_b' => [ 'case0-1', 'case1-1', 'case2-1', 'case3-1' ],
      'niv3-1_c' => { 'niv3-2_a' => 'test', },
    },
    'niv4' => \&FonctionBonjour,
  );

  sub FonctionBonjour {
    my $prenom = shift;
    print "Bonjour $prenom\n";
  }
```

Pour appeler la fonction FonctionBonjour via le hash, il faut écrire :

```
  # Afficher bonjour via le hachage
  $HashReferenceComplex{'niv4'}->('Djibril');
```

Autres affichages

```
  # Afficher bonjour via le hachage
  $HashReferenceComplex{'niv4'}->('Djibril');

  # Afficher valeur3-1
  print $HashReferenceComplex{'niv3'}{'niv3-1_a'}, "\n";

  # Afficher case0-1
  print $HashReferenceComplex{'niv3'}{'niv3-1_b'}->[0], "\n";

  # Afficher case0 à case3
  foreach my $niveau ( @{ $HashReferenceComplex{'niv2'} } ) {
    print "\t- $niveau\n";
  }
```

Cas particulier (tableau à deux dimensions) !

Auteurs : Djibril ,

Voici un exemple de tableau à deux dimensions (tableaux de tableaux).

```
  my @tab2tab = ([1,2], ["a","b"],["toto","tete"],["Djibril","Stoyak","vous"]);
```

Chaque case du tableau @tab2tab contient une référence à un tableau anonyme, c'est-à-dire une référence anonyme.

- **1re case du tableau @tab2tab => [1,2] ;**
- **2e case du tableau @tab2tab => ["a","b"] ;**
- **3e case du tableau @tab2tab => ["toto","tete"] ;**
- **4e case du tableau @tab2tab => ["Djibril","Stoyak","vous"] ;**
- **Petit rappel : un tableau normal s'écrit avec des parenthèses d'où @tab2tab = (...,...).**

un tableau anonyme avec des crochets : [1,2]
@tab2tab contient quatre références anonymes, ainsi pour récupérer la première référence anonyme :

```
  my $ref1 = $tab2tab[0]; # c'est-à-dire $ref1 = [1,2];
  my $ref4 = $tab2tab[3]; # c'est-à-dire $ref4 = ["Djibril","Stoyak","vous"];
  # =>Rappel : la première case d'un tableau a pour indice 0.
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Voici comment afficher la valeur "Stoyak", c'est-à-dire la deuxième case du 4e tableau anonyme de tab2tab :

```
my @tab2tab = ([1,2], ["a","b"],["toto","tete"],["Djibril","Stoyak","vous"]);
my $ref4 = $tab2tab[3];
```

```
print ${$ref4}[1];      #=> "Stoyak"

print $$ref4[1];        #=> "Stoyak"

print ${$tab2tab[3]}[1];#=> "Stoyak"
print $$tab2tab[3][1];  #=> ne fonctionnera pas (explication ci-dessous)

# Cas particulier des tableaux à plusieurs dimensions, on peut écrire simplement ainsi
print $tab2tab[3][1];   #=> "Stoyak"

print $tab2tab[3]->[1]; #ou print $ref4->[1]; => "Stoyak"
```

Explication 3) :
Je vous avais expliqué plus haut qu'il était possible d'omettre les accolades et vous avez dû vous demander, pourquoi ${$tab2tab[3]}[1]; est correct et pas $$tab2tab[3][1]; ?
En fait, pour Perl, pour des raisons de priorité entre opérateurs, $tab2tab[3] sera considéré comme une référence, qui n'existe pas. Vous aurez un message d'erreur à l'exécution du script. Si vous utilisez use strict, vous aurez :

```
Global symbol "$tab2tab" requires explicit package name at .............
```

Pour Perl, $tab2tab[3] est une référence qui n'existe pas car n'a pas été déclarée. Le fait de mettre des accolades permet clairement d'indiquer que la référence est $tab2tab[3]. Donc soit vous tapez :

```
print ${$tab2tab[3]}[1]; #=> "Stoyak"
```

ou bien

```
my $ref4 = $tab2tab[3]; #on déclare bien la référence $ref4
print $$ref4[1];        #=> "Stoyak" et c'est Ok.
```

J'espère avoir été clair !!!
Remarque : En général, dès que l'on parle de tableaux à deux dimensions ou plus, il faudra manipuler les références anonymes.

```
my @tab2tab = ([1,2], ["a","b"],["toto","tete"],["Djibril","Stoyak","vous"]);
print $tab2dimension[3][1];#=> Stoyak
```

Moyen mnémotechnique pour les tableaux à deux dimensions.
@tab2dimension => $tab2dimension[LIGNE][COLONNE]
Explication schématique de @tab2dimension

```
([1,2], ["a","b"],["toto","tete"],["Djibril","Stoyak","vous"]) correspond à

ligne0 -->   1      2
ligne1 -->   a      b
ligne2 -->   toto   tete
ligne3 -->   Djibril    Stoyak     vous
```

Exemple :
$tab2dimension[3][1] correspond à la ligne 3 et colonne 1, donc c'est Stoyak
première ligne => ligne0 et première colonne => colonne 0 (on est dans les tableaux !!!!!!)

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Voilà, vous maîtrisez maintenant les références !!! Je vais vous faire un petit résumé ci-dessous avec les pièges à éviter.

Astuces sur les références et erreurs à éviter !

Auteurs : Djibril ,

Si vous souhaitez savoir si $ref est une référence à un tableau ou un hash, voici une fonction Perl

```
my $ref_tableau = \@tableau;
my $ref_hash    = \%hash;
print ref $ref_tableau;    # ARRAY
print ref $ref_hash;       # HASH
```

Erreurs à éviter : Ne confondez pas $ref{'toto'} et $ref->{'toto'}.

- **$ref{'toto'} correspond à la valeur de la toto d'un hash nommé ref ;**
- **$ref->{'toto'} correspond à la valeur d'une référence nommée ref d'un hash.**

Prenez pour habitude de donner des noms simples et clairs à vos variables, et de bien commenter vos scripts !

- **Ne pas confondre un tableau (1,2,3) et une référence anonyme à un tableau anonyme [1,2,3] ;**
- **Ne pas confondre un hash ("cle" => "valeur") et une référence anonyme à un hash anonyme {"cle" => "valeur"}.**

Attention :

```
ma_fonction(@tab1, @tab2);
ou
ma_fonction(%hash1, %hash2);
```

ne fonctionnera pas. Utilisez des références en argument :

```
ma_fonction(\@tab1, \@tab2);
ou
ma_fonction(\%hash1, \%hash2);
```

Si vous souhaitez une documentation plus complète, vous pouvez taper perldoc perlref sous Windows ou man perlref sous Unix/Linux.

Qu'est-ce qu'une fermeture ?

Auteurs : Woufeil ,

Une fermeture est une référence anonyme vers une fonction. Ces références ont des propriétés très intéressantes, comme celle de garder en mémoire des valeurs de variables lexicales devenues hors de portée. Ainsi, en plus de définir les valeurs d'une fonction lors de l'appel, on peut aussi le faire lors de la définition. Un petit exemple :

```
my $ref;
{
 my $var = "Salut";
 $ref = sub {return "$var @_";};
}
print $ref->("Larry"); # affiche Salut Larry, alors que $var n'existe plus !
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Remarquez que l'on a défini une variable à la définition et une autre à l'appel.
On pourrait aussi passer $var à ferm et retourner $ref à la fin : à chaque appel de ferm on obtiendrait une fermeture (donc une référence vers un sous-programme) qui garderait la valeur de $var utilisée à l'appel de ferm. Cela donnerait ceci :

```
sub ferm {
  my ($var) = @_;
  my $ref = sub { return "$var @_"; };
  return $ref;
}
my $ref = ferm("Salut");

#plus tard
print $ref->("Larry");    #affiche Salut Larry
```

Comment générer automatiquement une fonction ?

Auteurs : Woufeil ,

En bouclant autour d'une fermeture, on peut créer des fonctions ayant un code semblable mais des noms différents. Le problème est qu'une fermeture n'a pas de nom, mais on peut le contourner : il est en effet possible de lier une référence de code à un nom déjà existant en passant par les typeglobs.
Par exemple, on veut deux fonctions fic1 et fic2 qui écrivent leurs paramètres dans des fichiers qui ont respectivement pour handle FIC1 et FIC2. Les deux fonctions sont très proches, elles ne diffèrent que par le fichier dans lequel écrire. Voilà comment générer ces deux fonctions en bouclant autour d'une fermeture :

```
foreach my $champ (qw(fic1 fic2))
{
 my $handle = uc $champ; # met $champ en majuscule
 no strict 'refs'; #Autorise les références symboliques
 *$champ = sub {print $handle "@_";}; # affecte la fermeture en question au typeglob
}
fic1("Salut !"); #écrit salut dans FIC1
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > Perl avancé > Opérateurs

Opérateur yada-yada

Auteurs : Djibril ,

Depuis la version Perl 5.12, un nouvel opérateur a vu le jour : l'opérateur yada-yada dont la syntaxe est trois petits points ...
Cet opérateur est aussi appelé "Whatever operator", "yada-yada operator" ou "et cetera operator".
C'est un marqueur très pratique qui permet de dire à Perl qu'il reste du code à écrire. Ainsi, lorsque Perl rencontre ce marqueur, il lance une exception dont le texte est le suivant : *Unimplemented at /home/djibril/test.pl line xx*. Il ne faut pas confondre cet opérateur avec un autre d'un nom aussi marrant "opérateur flip-flop".
Voici un exemple d'utilisation de yada-yada.
Vous avez un programme à concevoir, ce dernier contient plusieurs méthodes ou procédures. Si vous avez commencé à concevoir une procédure lister_fichiers, mais que vous ne l'avez pas terminée, vous pouvez y insérer ce marqueur. Ainsi, si vous faites appel à cette procédure accidentellement, Perl vous le signalera.

```
#!/usr/bin/perl
use warnings;
use 5.12.0;

afficher_bonjour('CLaudeLeLoup');
lister_fichiers('C:/');
afficher_bonjour('perl');


sub afficher_bonjour {
  my ($prenom) = @_;

  say "Bonjour $prenom";
}

sub lister_fichiers {
  my @arguments = @_;

  say "Procedure : lister_fichiers";
  # implémentation inachevée
  ...
}
```

```
Bonjour CLaudeLeLoup
Procedure : lister_fichiers
Unimplemented at C:\Documents and Settings\user\Bureau\test.pl line 23.
```

Pour en savoir plus, vous avez la documentation sur vos postes :

- **perldoc perlop (sous Windows)**
- **man perlop (Sous Linux ou MacOS)**
- **perlop CPAN**

defined-or

Auteurs : Djibril ,

Depuis Perl 5.10, il existe un opérateur provenant de Perl 6 s'appelant defined-or. Il est très pratique car permet d'écrire en peu de lignes une affectation de valeur par défaut. Vous allez rapidement comprendre en lisant un exemple.
Le code suivant :

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```perl
#!/usr/bin/perl
use warnings;
use strict;

my $prix;
my ($prix1, $prix2) = (1000, 2000);
if ( defined $prix1 ) {
  $prix = $prix1;
}
else {
  $prix = $prix2;
}
```

équivaut à

```perl
#!/usr/bin/perl
use warnings;
use strict;

my $prix;
my ($prix1, $prix2) = (1000, 2000);
$prix = defined $prix1 ? $prix1 : $prix2;
```

qui lui même équivaut à

```perl
#!/usr/bin/perl
use warnings;
use strict;

my $prix;
my ($prix1, $prix2) = (1000, 2000);
$prix = $prix1 // $prix2;
```

Smart match

Auteurs : Djibril ,

Depuis Perl 5.10, il existe un opérateur de comparaison nommé smart match dont la notation est : ~~
Il permet de comparer génériquement tout et n'importe quoi. C'est un opérateur booléen commutatif ($a ~~ $b équivaut à $b ~~ $a) très puissant importé de Perl 6. Son comportement dépend des arguments à comparer.

$a	$b	Type de correspondance	Code équivalent
Hash	Hash	Clefs de hash identiques (toutes les	

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

		clefs sont trouvées dans les deux hash)	
Hash	Tableau	Intersection des clés du hash	grep { exists $a->{$_} } @$b
Hash	CodeRef	sub truth for each key[1]	!grep { !$b->($_) } keys %$a
Hash	Regex	hash key grep	grep /$b/, keys %$a
Any	Hash	Existence d'une entrée de hash	exists $b->{$a}
Any	undef	Non définie	!defined $a
Any	Object	invokes ~~ overloading on $object, or dies	
Any	Tableau	match contre un élément du tableau	
Any	Regex	pattern match	$a =~ /$b/
Any	Num	égalité numérique	$a == $b
Any	CodeRef	scalar sub truth	$b->($a)
Any	Any	égalité des scalaires	$a eq $b
Tableau	Tableau	Tableau compatible	
Tableau	Hash	Intersection des clés du hash	grep { exists $b->{$_} } @$a
Tableau	CodeRef	sub truth for each elt [1]	!grep { !$b->($_) } @$a
Tableau	Regex	grep de tableau	grep /$b/, @$a
Regex	Hash	grep clef de hash	grep /$a/, keys %$b
Regex	Tableau	grep de tableau	grep /$a/, @$b
undef	Hash	always false (undef can't be a key)	
undef	Tableau	Tableau contenant un undef	grep !defined, @$b
undef	Any	Non définie	!defined($b)

[1] - Les listes et listes associatives matcheront.
Voici un exemple de programme :

```
#!/usr/bin/perl
use warnings;
use strict;
use 5.10.0;


my @fruits = qw< pomme orange kiwi fraise >;
my %personnes = (
  'paris'  => '75',
  'rennes' => '35',
  'reims'  => '51',
  'lyon'   => '69',
);
my $ref_hash = \%personnes;
my @villes = qw / paris rennes reims lyon/;
my $nombre1 = 12.35;
my $nombre2 = 12.35000;
if ( %personnes ~~  @fruits ) {
  print "\%personnes et \@fruits identiques\n";
}
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
if ( %personnes ~~  @villes ) {
  print "\%personnes et \@villes identiques\n";
}
if ( %personnes ~~  $ref_hash ) {
  print "\%personnes et \$ref_hash identiques\n";
}
if ( @villes ~~  @fruits ) {
  print "\@villes et \@fruits identiques\n";
}
if ( $nombre1 ~~  $nombre2 ) {
  print "\$nombre1 et \$nombre2 identiques\n";
}
```

Résultat

```
%personnes et @villes identiques
%personnes et $ref_hash identiques
$nombre1 et $nombre2 identiques
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > Perl avancé > Pragmas

Informations générales

Auteurs : Djibril ,

Depuis la version 5.12 de Perl, il y a maintenant beaucoup d'avertissements émis par défaut sur les fonctionnalités considérées obsolètes. Il existe une nouvelle catégorie deprecated que l'on peut désactiver via le code suivant :

```
no warnings 'deprecated';
```

Pragma 5.10.0

Auteurs : Djibril ,

En perl 5.10, faire appel au module 5.10.0 permet d'utiliser say au lieu de print par exemple.

```
#!/usr/bin/perl
use strict;
use warnings;
use 5.10.0;

say 'Bonjour';
```

Pragma 5.12.0

Auteurs : Djibril ,

Depuis perl 5.12, faire appel au module 5.12.0 charge implicitement le module strict et le feature ':5.12'. Les deux codes ci-dessous sont équivalents.

```
#!/usr/bin/perl
use warnings;
use 5.12.0;
```

```
#!/usr/bin/perl
use warnings;
use strict;
use feature ':5.12';
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > Programmation orientée objets en Perl

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > Programmation orientée objets en Perl > Introduction

Qu'est-ce que la Programmation Orientée Objet (POO) ?

Auteurs : Woufeil ,

La POO est une philosophie (bien que l'on dise plutôt un paradigme) de programmation reposant sur deux notions essentielles : l'abstraction et la classification. Elle se base sur la création de classes. Une classe est un type de données permettant de décrire ce que l'on appelle un objet à l'aide de variables et de méthodes. Ainsi, si nous devions définir une classe "humain", elle comprendrait sûrement les variables nom, prénom, sexe, nationalité... Une méthode est une fonction (ou une procédure) permettant de décrire un comportement. Par exemple, notre classe humain devrait comprendre les méthodes marcher, parler, manger, dormir... Un objet est donc une instance d'une classe. Par exemple, Mr Dupond est un objet de la classe humain. En tant que tel, il est donc doté des variables sexe (qui vaudra masculin pour lui), nom (Dupond), prénom (disons Jacques) et de toutes les autres variables définies dans la classe Humain. Il dispose également des méthodes propres à la classe humain, Mr Dupond peut donc parler, marcher, manger, dormir... Une fois conçue, une classe doit être considérée comme une boîte noire : on l'utilise sans se soucier de comment elle fonctionne, comme avec une télévision par exemple. C'est la notion d'abstraction.

Qu'est-ce que l'héritage de classe ?

Auteurs : Woufeil ,

L'héritage permet de définir des relations "est un" (IS A) entre plusieurs classes. C'est la notion de classification. Ainsi, on peut dire que l'humain est un mammifère, qui est lui-même un animal. En tant que mammifère, l'homme hérite des propriétés des mammifères, comme le fait d'avoir quatre membres, des poils... Il hérite aussi des méthodes de cette classe. Ainsi un homme comme tout mammifère peut bouger. A cet héritage la classe humain ajoute des particularités, comme la méthode "parler", ou la variable "nationalité". On dit que la classe humain hérite de la classe mammifère. Mammifère est la classe parente et humain est la classe fille. Comme un humain est un mammifère, on peut utiliser le premier à la place du second (et pas l'inverse, un mammifère n'étant pas nécessairement un humain). Ainsi, si un sous-programme attend comme argument un mammifère, on peut lui passer un humain sans qu'il n'y ait de problème.

Qu'est-ce que la surcharge de méthode ? Le Polymorphisme ?

Auteurs : Woufeil ,

Une classe fille peut redéfinir les méthodes d'une classe parente. Ainsi, la classe mammifère a une méthode se déplacer mais la classe cheval peut avoir envie de surcharger (redéfinir) cette méthode en précisant qu'un cheval peut marcher au pas, au trot et au galop. On dit que la méthode marcher de mammifère a été surchargée par celle de la classe cheval. Mais que se passe-t-il si une fonction qui attend un mammifère comme argument reçoit un cheval et que cette fonction appelle la méthode se déplacer ? Dans ce cas la méthode appelée sera la méthode redéfinie dans la classe cheval. Mais si la fonction reçoit un humain et que cette dernière classe n'a pas redéfini la méthode se déplacer, la méthode appelée sera celle de la classe mammifère. Cette propriété de choisir quelle méthode appeler en fonction de l'argument passé à la fonction s'appelle le polymorphisme.

Qu'est-ce que l'encapsulation ?

Auteurs : Woufeil ,

Enfin, une dernière propriété chère à la POO est l'encapsulation. Elle permet de définir que les membres d'une classe (variables et méthodes) sont privés ou publics. Un membre public est accessible à partir de n'importe quel point du

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

programme, un membre privé n'est utilisable qu'au sein de la classe elle-même. Il existe aussi la notion de membre protégé, ce dernier ne sera utilisable que dans la classe elle-même et dans ses classes filles.

Comment Perl implémente-t-il la POO ?

Auteurs : Woufeil ,

Depuis peu, Perl supporte la POO. Pour l'implémenter, les développeurs de Perl ont choisi de réutiliser un maximum de choses plutôt que d'en créer beaucoup de nouvelles. Ainsi, une classe est un paquetage (ou module), un objet est un référent et une méthode est une fonction (ou une procédure).

Les Objets en Perl, comment les définir ?

Auteurs : Woufeil ,

Un objet est une référence (vers une table de hachage le plus souvent), ou plutôt son référent. La distinction entre une référence vers un objet et l'objet lui-même étant souvent brouillée par les programmeurs Perl, on dit souvent qu'un objet est une référence. J'utiliserais donc cette métonymie de temps en temps. Le référent d'une référence vers une table de hachage pouvant contenir des données complexes (des scalaires, des tableaux, des tables de hachage, des tableaux de tableaux...) et les associer à une chaîne de caractères, il est très approprié pour contenir un objet. Par exemple, si l'on déclare $dupond comme objet de la classe humain (ou plutôt comme référence vers cet objet), $dupond->{nom} représente son nom, $dupond->parler() représente la méthode parler appliquée à l'objet dupond.

Quelle est la différence entre une méthode et une fonction d'un module ?

Auteurs : Woufeil ,

Une méthode est une fonction un peu particulière de par son premier argument (ou paramètre, c'est la même chose). Il existe deux types de méthodes :

- **les méthodes de classe qui s'appliquent à toute une classe (typiquement une méthode pour construire un objet de la classe). Une méthode de classe reçoit comme premier paramètre le nom du paquetage (c'est-à-dire le nom de la classe) utilisé ;**
- **les méthodes d'instance qui ne s'appliquent qu'à un objet de la classe (comme la méthode parler()). Une méthode d'instance reçoit en premier paramètre la référence vers l'objet avec lequel elle a été appelée (dans notre cas une méthode d'instance s'appliquant à l'objet dupond recevrait donc $dupond comme premier argument).**

Que sont les accesseurs et les mutateurs ?

Auteurs : Woufeil ,

Les accesseurs et les mutateurs sont des méthodes permettant respectivement de lire et de fixer la valeur d'une variable d'instance, ceci dans le but de respecter au mieux l'encapsulation.
En effet, le principe de l'encapsulation est qu'un objet ne devrait être accessible que par ses méthodes ; or il peut être nécessaire d'accéder aux variables d'instance ; on utilise pour cela les accesseurs et les mutateurs.
L'intérêt des accesseurs n'est pas énorme, mais celui des mutateurs est bien plus grand : on peut vérifier grâce à eux si la valeur que l'on veut affecter à la variable est correcte (par exemple pour l'âge d'une personne, si le nombre n'est pas négatif) et le cas échéant lever une exception.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Comment créer un objet en Perl ?

Auteurs : Woufeil ,

Un objet se construit à l'aide d'une fonction spéciale appelée constructeur. C'est une méthode de classe, elle recevra donc pour premier argument le nom de la classe à laquelle appartient l'objet. En Perl, un objet est un référent, qui est la plupart du temps une table de hachage. Pour déclarer ce référent comme un objet, on utilise la fonction bless (que l'on pourrait traduire "consacrer" en français) en lui passant comme argument la référence vers l'objet et le nom de la classe de l'objet (donc le nom du module contenant le constructeur). Cette fonction va lier la référence à la classe pour ensuite retourner la référence consacrée. Ainsi, l'objet "saura" à quelle classe il appartient. Un petit exemple vaut mieux qu'un long discours :

```
Package Classe1;
use strict;
sub constructeur
{
   my ($classe) = @_; #la fonction reçoit comme premier paramètre le nom de la classe
   my $this = {}; #référence anonyme vers une table de hachage vide
   bless ($this,$classe); #lie la référence à la classe
   return $this; #on retourne la référence consacrée
}
1; #indique la fin du module
```

Voilà un exemple de constructeur. Pour déclarer un objet membre de Classe1, il suffit d'appeler ce constructeur. On peut définir plusieurs constructeurs pour la même classe en Perl, il suffit de créer une seconde fonction retournant une référence consacrée.

Comment invoquer le constructeur ?

Auteurs : Woufeil ,

Pour invoquer des méthodes en Perl, on utilise l'opérateur flèche qui s'écrit comme ceci : ->. La syntaxe d'invocation est celle-ci :

```
invoquant->méthode(paramètres);
```

L'invoquant est soit le nom de la classe si l'on invoque une méthode de classe, soit le nom de l'objet auquel s'applique la méthode si l'on invoque une méthode d'instance. Ici, notre constructeur est une méthode de classe, son invoquant est donc le nom de la classe dans laquelle il est défini. Ainsi, pour créer un objet appelé obj1 à l'aide du constructeur précédemment défini, on fera ainsi :

```
my $obj1 = Classe1->constructeur();
```

"Mais pourquoi ne passons-nous pas comme paramètre à la fonction le nom de la classe ?" allez-vous me dire ? C'est fait automatiquement ! La méthode recevra toute seule comme premier paramètre l'invoquant par lequel elle a été

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

invoquée, ici Classe1. Il existe également une seconde syntaxe d'invocation : méthode invoquant (paramètres). On peut donc invoquer le constructeur comme ceci : constructeur Classe1();

Comment définir les variables caractérisant un objet ?

Auteurs : Woufeil ,

C'est facile ! Dans le constructeur on crée une référence anonyme vers une table de hachage. Cette table n'est pas nécessairement (et ne devrait pas être) vide ! Reprenons l'exemple de la classe humain. On veut que notre humain ait un nom et un âge. Voici le constructeur qui permet de faire ça :

```
Packge Humain;
use strict;
sub constructeur
{
   my ($classe) = @_; #la fonction reçoit comme premier paramètre le nom de la classe
   my $this = {''nom'' => ''Dupond'',
               ''age'' => 35}; #initialise les variables nom et age
   bless ($this,$classe); #lie la référence à la classe
   return $this; #on retourne la référence consacrée
}
1;
```

Comment initialiser les variables caractérisant l'objet ?

Auteurs : Woufeil ,

Le constructeur est une fonction, il ne reçoit pas forcément un seul paramètre. Ainsi si l'on veut définir l'âge et le nom de l'humain que l'on va créer lors de l'appel du constructeur, il suffit de lui passer comme argument ces deux données :

```
# Appel du constructeur :
my $dupond = Humain->constructeur("Dupond",35);
```

Définition du constructeur :

```
sub constructeur
{
   my ($classe, $nom, $age) = @_;
   my $this = {''nom'' => $nom,
               ''age'' => $age}; #initialise les variables nom et age
   bless ($this,$classe); #lie la référence à la classe
   return $this; #on retourne la référence consacrée
}
```

Attention, le premier paramètre reste le nom de la classe ! Le premier paramètre donné lors de l'appel correspond au second paramètre du constructeur.

Qu'est-ce qu'un destructeur ? Comment en créer ?

Auteurs : Woufeil ,

Un destructeur est une méthode particulière d'une classe. Elle sera invoquée à chaque fois que plus aucune référence ne pointera vers l'objet à détruire. Si par exemple vous avez créé une référence $dupond vers un objet et que vous affectez à $dupond la valeur undef, plus aucune référence ne pointera vers l'objet $dupond et le destructeur sera appelé car $dupond ne sera plus accessible par aucun moyen. Le destructeur sert à détruire un objet correctement.
En Perl, le destructeur est une méthode d'instance nommée DESTROY. Exemple de destructeur.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```perl
#Dans le module Humain
sub DESTROY
{
   my ($this) = @_
   print "L'objet $dupond->{nom} va être détruit !";
}
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Comment accéder aux champs de l'objet créé ?

Auteurs : Woufeil ,

Vous avez déclaré un objet $dupond comme membre de la classe Humain, très bien. Maintenant il faudrait pouvoir avoir accès aux différents champs le concernant (son nom, son prénom, son âge...). Souvenez-vous que $dupond est une référence vers une table de hachage, il faut donc utiliser l'opérateur flèche pour accéder à ses différents champs. Par exemple, pour accéder au champ age de $dupond, il faut faire :

```
$dupond->{age}
```

Et si la variable est un tableau tab ?

```
$dupond->{tab}->[1];
#accède au second élément de @tab  . Equivalent à
$dupond->{tab}[1];
```

De même pour une table de hachage.

Comment créer une méthode d'instance en Perl ?

Auteurs : Woufeil ,

En Perl, une méthode n'est qu'une fonction recevant un premier paramètre un peu spécial. Une méthode d'instance recevra comme premier argument la référence vers l'objet sur lequel elle doit s'appliquer.
Supposons que cette référence pointe vers une table de hachage (c'est le cas la plupart du temps) et définissons une méthode defage permettant de définir l'âge de l'humain passé à cette méthode et de l'afficher :

```
sub defage
{
   my ($this, $age ) = @_;
   $this->{age} = $age;
   print ''$this->{prenom} $this->{nom} a $this->{age} ans'';
}
```

Il faut définir cette méthode dans le paquetage contenant la classe bien entendu !

Comment appeler une méthode d'instance ?

Auteurs : Woufeil ,

C'est pratiquement la même chose que pour appeler un constructeur ! Supposons que nous ayons créé un objet $dupond membre de la classe Humain via le constructeur associé. Si maintenant on veut appeler la méthode defage sur cet objet il suffit de taper :

```
$dupond->degafe(35);
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Comme un appel de méthode lui passe (à la méthode) de lui-même son invoquant en tant que premier paramètre, il ne faut pas passer la référence $dupond. Ainsi, la fonction defage recevra bien $dupond comme premier paramètre et 35 en second. Ce n'est pas plus difficile que ça !

Générer automatiquement des accesseurs/mutateurs (version AUTOLOAD)

Auteurs : Woufeil ,

On peut utiliser la fonction AUTOLOAD pour charger une méthode non définie. On peut s'en servir pour facilement générer des accesseurs et des mutateurs. Cela se fait ainsi :

```
sub AUTOLOAD {
  our $AUTOLOAD;
  return if $AUTOLOAD =~ /DESTROY/;     # vérifie que la méthode n'est pas le destructeur
  my $this  = shift;                            # $this contient une référence à l'objet courant
  my $champ = __PACKAGE__ . "::" . $AUTOLOAD;
  die "Champ invalide" unless exists $self->{$champ};     # vérifie que le champ existe
  if (@_) { $this->{$champ} = shift; }
  # on affecte la nouvelle valeur de champ à champ
  return $self->{$champ};                                 # retourne la valeur du champ considéré.
}
```

Maintenant comment utiliser cette méthode via un objet $dupond ayant un attribut nom ?

```
$dupond->nom(); #retourne Dupond;
$dupond->nom("Jean Dupond"); #affecte Jean Dupond à nom
```

Générer automatiquement des accesseurs/mutateurs (version fermetures)

Auteurs : Woufeil ,

Les fermetures sont des puissants générateurs de fonctions pour peu que l'on boucle autour. On peut donc s'en servir pour générer des accesseurs/mutateurs du même type que ceux de la version AUTOLOAD :

```
foreach my $iteration ( "nom", "prenom", "age" ) {
  my $nomcomplet = __PACKAGE__ . "::$iteration";
  no strict 'refs';    # pour l'instruction suivante
  *$nomcomplet = sub {
    my $this = shift;
    $this->{$iteration} = shift if (@_);
    return $this->{$iteration};
  };
}
```

Le principe est simple : mettez dans le foreach chaque champ pour lequel vous souhaitez qu'il dispose d'un accesseur et d'un mutateur. Rien ne change en ce qui concerne l'utilisation :

```
$dupond->nom(); # retourne Dupond;
$dupond->nom("Jean Dupond"); # affecte Jean Dupond à nom
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Comment une classe hérite-t-elle en Perl ?

Auteurs : Woufeil ,

Pour faire un héritage de classe en Perl, il suffit de créer dans la classe qui hérite un tableau nommé @ISA qui contiendra la ou les classes dont elle hérite ! Ce n'est pas plus compliqué que ça. Ce tableau sera déclaré avec our et non my. Le constructeur d'une telle classe contiendra un appel au(x) constructeur(s) de la (ou des) classe(s) dont elle hérite. Il obtiendra donc une référence consacrée, à laquelle il pourra rajouter ses propres variables, il consacrera à nouveau cette référence (il la liera avec la classe courante bien entendu) pour enfin la retourner.
Imaginons que le la classe Humain hérite d'une classe Animal qui définit les variables nom et age. A ces variables la classe Humain rajoutera la variable nationalité. On suppose que le constructeur de la classe Animal reçoit comme paramètre le nom et l'âge de l'animal à créer. Définissons le constructeur de la classe Humain qui recevra comme paramètre la nationalité de l'humain à créer.

```
package Humain;
use strict;
use Animal;    # Humain se servant des variables et méthodes de Animal, il faut inclure Animal
our @ISA = ("Animal");    # Le fameux tableau...

sub consthumain {
  my ( $classe, $nat ) = @_;

  #la ligne suivante appelle le constructeur d'Animal
  my $this = $classe->SUPER::constanimal( "dupond", 35 );
  $this->{nationalite} = $nat;    # crée un champ age et lui affecte $nat
  bless $this, $classe;           # lie la référence à la classe courante
  return $this;
}
```

Le SUPER:: demande à Perl d'aller chercher la méthode constanimal dans l'une des classes mères de Humain. Le constructeur sera appelé avec, comme invoquant, le nom de la classe en question. Pour invoquer les constructeurs, il suffit de faire :

```
my $dupond = Humain->consthumain("française");
```

Comment mettre en oeuvre le polymorphisme en Perl ?

Auteurs : Woufeil ,

Pour ça, Perl vous simplifie la vie. En C++ il faudrait déclarer des fonctions virtuelles, ici nul besoin de faire quoi que ce soit, Perl choisira automatiquement la méthode la plus appropriée en fonction de l'invoquant. Comment ? Perl cherchera tout d'abord dans la classe de l'invoquant si la méthode invoquée existe. Si ce n'est pas le cas, Perl prendra le premier membre du tableau @ISA (qui sera donc une classe mère) et cherchera s'il existe une méthode correspondante dans cette classe, et ainsi de suite avec les autres classes mères en cas d'héritage multiple. Rappelez-vous du premier

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

paramètre des méthodes et du fait qu'une référence "sait" à quelle classe appartient l'objet vers lequel elle pointe. Cela permet de comprendre comment le polymorphisme est mis en oeuvre aussi facilement.

Peut-on faire de l'héritage multiple ? Comment ?

Auteurs : Woufeil ,

Oui, bien sûr ! Rappelez-vous, @ISA est un tableau, il ne contient pas forcément qu'un seul élément. Tous les éléments de @ISA sont les classes mères de la classe dans laquelle ce tableau est déclaré.

Héritage et destructeur

Auteurs : Woufeil ,

Il y a un petit problème lors de la destruction d'un objet dont la classe hérite d'autres classes en Perl. En effet, dans ce cas, seul le destructeur de l'objet est appelé alors que normalement les destructeurs des classes mères devraient l'être également. La solution consiste à faire un appel direct du destructeur des classes mères dans le destructeur de la classe fille. Par exemple, si le destructeur reçoit une référence $this comme premier paramètre, l'appel se fera ainsi :

```
$this->SUPER::DESTROY();
```

L'opérateur ::

Auteurs : Woufeil ,

Parlons déjà des portées des variables de module. Si dans un module Mod1 vous avez créé une variable $var que vous avez déclarée avez my, elle ne sera accessible qu'au sein même du module. Mais si vous l'avez déclarée avec our, elle le sera aussi à l'extérieur du module. Comment y accéder ? Avec l'opérateur :: et en utilisant cette syntaxe :

```
$Mod1::var;
#Attention, Mod1::$var est incorrect !
```

Imaginons que vous ayez défini une fonction fn() à l'intérieur de ce module. Comment appeler cette fonction ? Pareil que pour les variables :

```
Mod1::fn();
```

En généralisant, l'opérateur :: s'utilise ainsi :

```
<nom du module>::<nom de la variable ou de la fonction. Le symbole du type de variable sera rajouté
 au début de l'instruction>
```

Ce n'est pas plus compliqué que ça !

A quoi sert la dernière ligne d'un module ?

Auteurs : Woufeil ,

Vous avez sûrement remarqué que les modules se terminent souvent par un étrange 1. Comme vous le savez, 1 est une valeur "vrai", à l'inverse de 0 ou de undef. Lors de l'exécution de la directive use, le code du module est exécuté sauf

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

si Perl rencontre une valeur "faux" suivie d'un point virgule. Une valeur "faux" indiquera qu'il y a eu une erreur lors du chargement du module, un "vrai" que tout s'est bien passé, d'où l'ajout du 1; à la fin de chaque module.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > Gestion des dates

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Définition de Epoch

Auteurs : Djibril ,

En ce qui concerne la date *Epoch*, elle varie en fonction des systèmes d'exploitation.
L'Epoch (de l'anglais époque ou ère) représente la date initiale à partir de laquelle est mesuré le temps par les systèmes d'exploitation.

- **sous UNIX, c'est 1er janvier 1970 à 0 heure (UTC) ;**
- **sous Mac OS, le 1er janvier 1904 à 0 heure ;**
- **sous Mac OS X (étant basé sur UNIX, il utilise l'Epoch UNIX) donc 1970 à 0 heure ;**
- **sous VMS, le 1er janvier 1858 à 0 heure ;**
- **sous Windows le 1er janvier 1900 à 0 heure.**

Le temps est mesuré en nombre d'unités de temps depuis cette date. L'unité de temps la plus courante est la seconde. Il arrive que l'unité ne soit pas spécifiée et que l'on parle de tick, qui est la plus petite unité de temps gérée par le système d'exploitation.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sous Windows, Linux et Mac

Auteurs : Djibril ,

Pour voir la liste des éditeurs de texte les plus utilisés par les perléens, regardez notre page outils

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > Gestion des dates > Je ne trouve pas mes réponses

Où trouver d'autres réponses à mes questions ?

Auteurs : Djibril ,

Si vous ne trouvez pas vos réponses dans cette FAQ, n'hésitez pas à consulter les codes disponibles dans notre rubrique Perl, sans oublier les tutoriels et le forum Perl.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > Codes sources utiles

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > Codes sources utiles > Des codes sources

A quoi sert cette section ?

Auteurs : Djibril ,

Cette section a pour but de vous lister des codes sources utiles pour vos développements. Elle sert de listes des différents codes perl nécessitant quelques explications.

Si vous recherchez des codes sources prêts a être téléchargés, regardez la section " Comment télécharger ou uploader un code dans la rubrique Perl".

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > Codes sources utiles > Des codes sources > Bioinformatique

Comment contribuer à cette liste de questions réponses ?

Auteurs : Djibril ,

Si vous souhaitez :

- **participer à l'amélioration des codes en bioinformatique de cette source ;**
- **de soumettre des corrections ou toutes idées, d'amélioration ;**
- **de contribuer.**

N'hésitez pas, faites vos propositions ici.

Comment récupérer (proprement) les séquences d'un fichier fasta ?

Auteurs : Jasmine80 ,

Le but est de récupérer les identifiants et leur séquence une à une de façon simple et rapide grâce au module Bio::SeqIO. Fichier d'entrées :

test.txt

```
>A1
GATACCAGCATCGTACGTCGTACGTACGTAGGGATCATTATGCCACATTCTGATCTTGGACCTGCATTATAGATCTGACTT
>A2
TACCACCCGATCTCGCATCGTCATGTGCGGGATCATTATGCACATTCTGATCTTGGACCTGCATTATAGATCTGACTT
>B1
GATACCAGCGGGATCATTATGCCACATTCTGATCTTGGACCTGATAGATCTGACT
>B2
GATACCAGCCACTTCTGACGATCGATCGATATTATAAAAGGATCATTATGCCACATTCTGATCgTGGACCTGCATTATAGATCTGCCCTT
>C1
GATACCAGCGGGATCATTATGCCACATTCTGATCTTGGACCTGCATTATAGATCTGACTT
>C2
GATACCAGCGGGATCCTTATGCCACATTCTGATCTTGGACCTGCATTATAGATCTGACTT
```

```perl
#!/usr/bin/perl
use strict;
use warnings;
use Bio::SeqIO;

# fichier d'entrée
my $in = Bio::SeqIO->new( -file => "test.txt", '-format' => 'Fasta' );

# fichier de sortie
my $out = Bio::SeqIO->new( -file => ">exit.fas", '-format' => 'Fasta' );

# récupération des séquences
while ( my $seq = $in->next_seq() ) {
  my $id       = $seq->primary_id;
  my $sequence = $seq->seq;

  # écriture dans le fichier de sortie
  $out->write_seq($seq);
}
```

Résultat

```
>A1
GATACCAGCATCGTACGTCGTACGTACGTAGGGATCATTATGCCACATTCTGATCTTGGA
CCTGCATTATAGATCTGACTT
>A2
TACCACCCGATCTCGCATCGTCATGTGCGGGATCATTATGCACATTCTGATCTTGGACCT
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Résultat

```
GCATTATAGATCTGACTT
>B1
GATACCAGCGGGATCATTATGCCACATTCTGATCTTGGACCTGATAGATCTGACT
>B2
GATACCAGCCACTTCTGACGATCGATCGATATTATAAAAGGATCATTATGCCACATTCTG
ATCgTGGACCTGCATTATAGATCTGCCCTT
>C1
GATACCAGCGGGATCATTATGCCACATTCTGATCTTGGACCTGCATTATAGATCTGACTT
>C2
GATACCAGCGGGATCCTTATGCCACATTCTGATCTTGGACCTGCATTATAGATCTGACTT
```

Est-il possible de récupérer des sous-séquences d'un alignement ?

Auteurs : Jasmine80 ,

Le module Bio::AlignIO permet de récupérer et d'analyser les séquences une à une en gardant les positions des gaps. On peut donc aussi récupérer un bloc de sous-séquences en gardant leur alignement, mais également obtenir la séquence consensuelle de cet alignement.

file.fsa

```
>A1/1-60
GATACCAGCGGGATCATTATGCCACATTCTGATCTTGGACCTGCATTATAGATCTGACTT
.
>A2/1-57
..TACCAGCGGGATCATTATGC.ACATTCTGATCTTGGACCTGCATTATAGATCTGACTT
.
>B1/1-55
GATACCAGCGGGATCATTATGCCACATTCTGATCTTGGACCTG....ATAGATCTGACT.
.
>C1/1-60
GATACCAGCGGGATCATTATGCCACATTCTGATCTTGGACCTGCATTATAGATCTGACTT
.
>C2/1-60
GATACCAGCGGGATCCTTATGCCACATTCTGATCTTGGACCTGCATTATAGATCTGACTT
.
>B2/1-61
GATACCAGCCGGATCATTATGCCACATTCTGATCGTGGACCTGCATTATAGATCTGCCCT
T
```

```perl
#!/usr/bin/perl
use strict;
use warnings;

use Bio::AlignIO;

# fichier d'entrée
my $inputfilename = 'file.fsa';
my $in            = Bio::AlignIO->new(
  -file     => $inputfilename,
  '-format' => 'fasta'
);

my $aln = $in->next_aln();

# recherche de la séquence consensus à 50%
print "sequence consensus à 50%\n";
print $aln->consensus_string(50), "\n\n";

# manipulation de séquences alignées
foreach my $seq ( $aln->each_seq ) {

  # récupération de sous-séquences
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
    my $res = $seq->subseq( 10, 45 );
    print $res, "\n";
}
```

Résultat

```
sequence consensus à 50%
GATACCAGCGGGATCATTATGCCACATTCTGATCTTGGACCTGCATTATAGATCTGACTT?

GGGATCATTATGCCACATTCTGATCTTGGACCTGCA
GGGATCATTATGC.ACATTCTGATCTTGGACCTGCA
GGGATCATTATGCCACATTCTGATCTTGGACCTG..
GGGATCATTATGCCACATTCTGATCTTGGACCTGCA
GGGATCCTTATGCCACATTCTGATCTTGGACCTGCA
CGGATCATTATGCCACATTCTGATCGTGGACCTGCA
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > Codes sources utiles > Des codes sources > Expressions régulières

Formatage d'un nombre entier avec des espaces comme séparateurs des milliers

Auteurs : kuzco ,

Voici une fonction permettant le formatage d'un nombre entier avec des espaces comme séparateurs des milliers.

```
sub FormatNumber {
    my( $Number ) = @_;
    while( $Number =~ s/^(-?\d+)(\d{3})/$1 $2/ ){};
    return( $Number );
}
```

Exemple d'utilisation :

```
FormatNumber(95412368);
Retourne : 95 412 368
```

Comment trouver le nombre d'occurrence d'un motif dans une chaine ?

Auteurs : Djibril , Philou67430 ,

```
#!/usr/bin/perl
use strict;
use warnings;

my $motif = 'e';
my $phrase = 'Comment lire un fichier de configuration (.ini, yaml, ...) ?';
my $count =()= $phrase =~ m{$motif}g;
print "Il y a $count \"$motif\" dans \$phrase\n";
```

Résultat

```
Il y a 4 "e" dans $phrase
```

=()= est ce qu'on appelle en Perl un goatse, pas toujours connu du grand public.
En fait, c'est l'usage de l'opérateur d'affectation à une liste vide qui permet d'évaluer l'expression régulière dans un contexte de liste et dont le résultat est affecté à $count. Or la valeur de retour d'une affectation de liste évaluée dans un contexte de scalaire retourne le nombre d'élément contenu dans la rvalue de cet opérateur d'affectation, donc le nombre d'élément capturé par l'expression régulière.

Comment mettre la première lettre de tous les mots d'une chaine en majuscule ou minuscule ?

Auteurs : Djibril ,

L'utilisation des expressions régulière est plus appropriée.

```
#!/usr/bin/perl
use strict;
use warnings;

my $motif = 'e';
my $phrase = 'Comment lire un fichier de configuration (.ini, yaml, ...) ?';
print "$phrase\n";

# tous les mots en majuscule
$phrase =~ s{\b(.+?)\b}{ucfirst($1)}ge;
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
print "Maj : $phrase\n";

# tous les mots en minuscule
$phrase =~ s{\b(.+?)\b}{lcfirst($1)}ge;
print "Min : $phrase\n";
```

Résultat

```
Comment lire un fichier de configuration (.ini, yaml, ...) ?
Maj : Comment Lire Un Fichier De Configuration (.Ini, Yaml, ...) ?
Min : comment lire un fichier de configuration (.ini, yaml, ...) ?
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Comment lister les fichiers d'un répertoire ?

Auteurs : Djibril ,

Le but est de pouvoir lister les fichiers d'un répertoire de manière récursive ou non. Voici une procédure que vous permettra de la faire. Le premier argument doit être un nom de répertoire, et le deuxième (optionnel) doit être 1 (recherche recursive) ou 0 (pas de rechercher récursive).

Exemple de script pour lister les fichiers

```
#!/usr/bin/perl
use warnings;
use strict;

my $repertoire = 'C:/tmp';
foreach my $fichier ( lister_fichiers( $repertoire, 1 ) ) {
  print "Fichier : $fichier\n";
}

#=======================================================
# Nombre d'arguments : 1 ou 2
# Argument(s)        : un répertoire et valeur 0 ou 1
# Retourne           : Tableau de fichier (@fichiers)
#=======================================================
sub lister_fichiers {
  my ( $repertoire, $recursivite ) = @_;
  require File::Spec;

  # Recherche dans les sous répertoire ou non
  if ( ( not defined $recursivite ) || ( $recursivite != 1 ) ) { $recursivite = 0; }

  # Verification répertoire
  if ( not defined $repertoire ) { die "Aucun repertoire de specifie\n"; }

  # Ouverture d'un répertoire
  opendir my $fh_rep, $repertoire or die "impossible d'ouvrir le répertoire $repertoire\n";

  # Liste fichiers et répertoire sauf (. et ..)
  my @fic_rep = grep { !/^\.\.?$/ } readdir $fh_rep;

  # Fermeture du répertoire
  closedir $fh_rep or die "Impossible de fermer le répertoire $repertoire\n";

  # On récupère tous les fichiers
  my @fichiers;
  foreach my $nom (@fic_rep) {
    my $notre_ficrephier = File::Spec->catdir( $repertoire, $nom );

    if ( -f $notre_ficrephier ) {
      push( @fichiers, $notre_ficrephier );
    }
    elsif ( -d $notre_ficrephier and $recursivite == 1 ) {
      push( @fichiers, lister_fichiers($notre_ficrephier, $recursivite) );    # recursivité
    }
  }
  return @fichiers;
}
```

Résultat

```
Fichier : C:\tmp\.metadata\.log
Fichier : C:\tmp\.metadata\.plugins\org.eclipse.core.runtime\.settings
\com.sneda.pgi.application.prefs
Fichier : C:\tmp\.metadata\.plugins\org.eclipse.core.runtime\.settings\org.eclipse.ui.prefs
Fichier : C:\tmp\.metadata\.plugins\org.eclipse.ui.workbench\dialog_settings.xml
Fichier : C:\tmp\.metadata\.plugins\org.eclipse.ui.workbench\workbench.xml
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Cette procédure utilise le module File::Spec qui permet d'avoir les bons chemins de fichiers quelque soit la plateforme.
Si vous chercher un module pour lister des fichiers et vous permettant d'effectuer de traitements sur chacun d'eux, regardez le module File::Find. Il est présent dans le core de Perl, il est inutile de chercher à l'installer. Exemple :

```
#!/usr/bin/perl
use warnings;
use strict;
use File::Find;

find( { wanted => \&process, }, 'C:/tmp' );

sub process {
  if ( -f $File::Find::name ) { print "File::Find : $File::Find::name\n"; }
}
```

Résultat

```
File::Find : C:\tmp\.metadata\.log
File::Find : C:\tmp\.metadata\.plugins\org.eclipse.core.runtime\.settings
\com.sneda.pgi.application.prefs
File::Find : C:\tmp\.metadata\.plugins\org.eclipse.core.runtime\.settings\org.eclipse.ui.prefs
File::Find : C:\tmp\.metadata\.plugins\org.eclipse.ui.workbench\dialog_settings.xml
File::Find : C:\tmp\.metadata\.plugins\org.eclipse.ui.workbench\workbench.xml
```

Comment utiliser des noms de fichiers ou répertoires portables ?

Auteurs : Djibril ,

Lorsque vous manipulez des fichiers ou répertoires, vous avez souvent besoin de construire et manipuler des chemins. Le premier réflexe sous Linux est d'utiliser des slashes (/) ou des backslashes (\) sous Windows. Des fois, vous pouvez être amené à utiliser des expressions régulières pour convertir des / en \ ou l'inverse.
Ce genre de manipulation est assez fastidieux et ne rend pas toujours les programmes portables. Voici un exemple :

```
#!/usr/bin/perl
use strict;
use warnings;

my $repertoire     = 'C:/repertoire';
my $nom_fichier    = 'fichier.txt';
my $chemin_fichier = $repertoire . '/' . $nom_fichier;
```

ou

```
#!/usr/bin/perl
use strict;
use warnings;

my $repertoire     = 'C:\repertoire';
my $nom_fichier    = 'fichier.txt';
my $chemin_fichier = $repertoire . '\' . $nom_fichier;
```

La meilleure solution est de toujours utiliser le module Perl File::Spec qui est disponible dans le core de Perl. Vous n'aurez plus besoin de vous soucier de la plateforme sous laquelle tournera le programme. Voici un exemple d'utilisation :

```
#!/usr/bin/perl
use strict;
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
use warnings;
use File::Spec;

my $repertoire     = 'C:\repertoire';
my $nom_fichier    = 'fichier.txt';
my $chemin_fichier = File::Spec->catfile($repertoire, $nom_fichier);
print $repertoire,"\n";

my $chemin_repertoire = File::Spec->catdir('\test', 'tmp', 'dossier');
print $chemin_repertoire,"\n";
```

Résultat sous Windows

```
C:\repertoire
\test\tmp\dossier
```

Résultat sous Linux/Mac

```
C:/repertoire
/test/tmp/dossier
```

Pour en savoir plus, consultez la documentation officielle du module.

lien : File::Spec

Comment renommer ou copier un fichier ?

Auteurs : Djibril ,

- **Renommer un fichier**

Pour renommer un fichier en perl, une fonction perl existe déjà.

```
#!/usr/bin/perl
use strict;
use warnings;
use File::Copy;
# Pour renommer un fichier toto.txt en tutu.txt
rename("toto.txt","tutu.txt");
```

- **Copier un fichier**

Pour copier un fichier, la façon la plus propre et la plus simple est d'utiliser le module CPAN File::Copy déjà fait pour ça. C'est un module parmi tant d'autres.

```
#!/usr/bin/perl
use strict;
use warnings;
use File::Copy;
my $fichier1 = "toto.txt";
my $fichier2 = "tutu.txt";
# Copie le fichier dans le même répertoire avec un nouveau nom
copy($fichier1,$fichier2);

# copie le fichier dans un autre répertoire avec un nom différent
copy($fichier1,'C:/tata.txt');

# copie le fichier dans un autre répertoire avec le même nom
copy($fichier1,'C:/');
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Pour en savoir plus : File::Copy

Comment copier ou supprimer un répertoire en perl ?

Auteurs : Djibril ,

- **Faire une copie d'un répertoire**

Utiliser le module File::Copy::Recursive, il est fait pour ça.

```
#!/usr/bin/perl
use strict;
use warnings;
use File::Copy::Recursive qw(fcopy rcopy dircopy fmove rmove dirmove);
my $repertoire1 = "./rep1";
my $repertoire2 = "./rep2";
dircopy($repertoire1,$repertoire2) or die("Impossible de copier $repertoire1 $!");
```

- **Supprimer un répertoire en perl**

Il existe une fonction en perl permettant de supprimer un répertoire en perl, mais ce dernier ne fonctionne que si le répertoire en question est vide.

```
#!/usr/bin/perl
use strict;
use warnings;

my $repertoire_vide = "./rep1";
rmdir($repertoire);
```

Si vos répertoires ne sont pas vides, vous pouvez parcourir tous le repertoire récursivement et supprimer les fichiers. Ensuite parcourir les répertoires un à un et les supprimer un à un en partant de celui le plus en profondeur dans l'arborescence. Ca peut être un bon exercice. Mais sachez qu'il existe un module prêt à l'emploi et simple d'utilisation. Utiliser le module File::Path.

```
#!/usr/bin/perl
use strict;
use warnings;
use File::Path;
my $repertoire1 = "./rep1";
rmtree([$repertoire1, 1, 1);
```

En savoir plus sur File::Path

Comment créer un fichier temporaire proprement ?

Auteurs : Djibril ,

Il est souvent utile de créer un fichier temporaire lorsque l'on traite un fichier. On a pour mauvaise habitude de créer un fichier avec un nom arbitraire, de tester si ce dernier n'existe pas, etc. Bien évidemment, c'est une mauvaise idée car perl nous fournit déjà ce qu'il faut. (le module File::Temp est déjà dans le core de perl depuis perl 5.6.1).
Exemple :

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```perl
#!/usr/bin/perl
use strict;
use warnings;
use File::Temp qw/ tempfile /;

# Création d'un fichier temporaire qui sera supprimé une fois le script terminé.
# N'oubliez pas les 4 XXXX, perl créera un fichier avec à la place des X des caractères aléatoires.
my ($fh_temp, $file_temp) = tempfile("fichier_temporaireXXXX", UNLINK => 1);

# Fermeture du fichier temporaire
close($fh_temp);
```

Comment récupérer le nom (ou chemin) ou l'extension d'un fichier?

Auteurs : Djibril ,

Il y a deux solutions :

- **utiliser les expressions régulières**

```perl
#!/usr/bin/perl
use strict;
use warnings;
my $fichier = "C:\\Documents and Settings\\Djibril\\fichier_faqperl.txt";
#ou my $fichier = "C:/Documents and Settings/Djibril/fichier_faqperl.txt";
#ou my $fichier = "/home/Djibril/fichier_faqperl.txt";

my ($repertoire,$nom_fichier) = $fichier =~ /(.+[\/\\])([^\/\\]+)$/;
print "($repertoire,$nom_fichier)\n";
```

- **utiliser File::Basename**

```perl
#!/usr/bin/perl
use strict;
use warnings;
use File::Basename;
my $fichier = "C:\\Documents and Settings\\Djibril\\fichier_faqperl.txt";
#ou my $fichier = "C:/Documents and Settings/Djibril/fichier_faqperl.txt";
#ou my $fichier = "/home/Djibril/fichier_faqperl.txt";

my $repertoire = dirname($fichier);
my $nom_fichier = basename($fichier);
print "($repertoire,$nom_fichier)\n";
```

Pour récupérer également l'extension d'un fichier, le module File::Basename nous facilite grandement la vie grâce à la méthode fileparse. Voici des exemples de codes provenant de la documentation du module.

- **fileparse**

```perl
my ($filename, $directories, $suffix) = fileparse($path);
my($filename, $directories, $suffix) = fileparse($path, @suffixes);
my $filename                         = fileparse($path, @suffixes);

fileparse("/foo/bar/baz.txt", qr/\.[^.]*/); # On Unix returns ("baz", "/foo/bar", ".txt")
fileparse("/foo/bar/baz");                  # On Unix returns ("baz", "/foo/bar/", "")
fileparse("C:/foo/bar/baz");     # On Windows returns ("baz", "C:/foo\bar/", "")
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
fileparse("/foo/bar/baz/");     # On Unix returns ("", "/foo/bar/baz/", "")
```

Modifier un fichier préexistant

Auteurs : Djibril , Jedai ,

Pour les débutants en programmation, sachez que le problème de modification d'un fichier et les solutions exposées ici sont utilisables dans n'importe quel langage. On ne peut pas simplement *"modifier une ligne"* dans un fichier, ceci est rendu impossible par la façon dont est stockée un fichier. En effet le fichier est une suite d'octets contigus en mémoire (ou du moins qui nous apparaissent ainsi à notre niveau), donc *"modifier une ligne"* implique forcément de déplacer toute la fin du fichier pour combler le trou à partir du moment où la modification implique un changement de longueur de la ligne...
Voici plusieurs stratégies pour traiter ce problème :

1. **On ouvre en lecture/écriture, on met l'ensemble du fichier en mémoire (par exemple dans un tableau de lignes), on le modifie là, on tronque le fichier initial et on réécrit la version modifiée**
2. **On lit enregistrement par enregistrement (*ligne par ligne* par exemple) le fichier, et on modifie chaque enregistrement selon ses besoins, puis on le réécrit dans un autre fichier, finalement on écrase le fichier original par la version modifiée.**
3. **Variation 1 : On ne met en mémoire que la fin du fichier, la partie à partir de la première modification**
4. **Variation 2 : On utilise un tampon pour mettre à jour la fin du fichier en procédant par petit bout, sans écraser les données qu'on n'a pas encore mis en mémoire.**

Chacune de ses solutions a ses avantages et ses inconvénients :

1. **Pour les gros fichiers cette stratégie est très lourde en mémoire.**
2. **Bonne solution, sauf si son disque est encombré (en effet le fichier est *présent en double sur le disque avant l'étape finale*)... De plus ce n'est pas forcément optimal de tout recopier alors qu'on a modifié qu'un enregistrement.**
3. **Pas mal, mais même problème que le 1 si l'enregistrement est au début du fichier.**
4. **L'idéal, ou presque, mais très complexe à mettre en place, surtout de façon intelligente (c'est à dire par exemple en permettant de commander plusieurs modifications avant de commencer à changer réellement le fichier sous-jacent).**

Nous vous conseillons de s'en tenir à la solution 2 dès lors que l'on a de grosses modifications à effectuer sur des fichiers dont on n'est pas sûr de la taille.
La solution 1 peut convenir si on est sûr que les fichiers resteront petits (mais on peut rarement être vraiment sûr dans un environnement informatique).
La solution 4 est trop complexe pour être utilisée à la main de façon raisonnable (efficacement), mais ça tombe bien, Perl est fourni en standard depuis la 5.8 avec un module Tie::File qui implémente ce modèle de façon intelligente. (Tie::File est compatible avec Perl depuis la version 5.4 mais n'est pas en standard avec les versions inférieures à la 5.8) Tie::File permet de "lier" un tableau à un fichier, de sorte que l'on peut manipuler un fichier comme s'il s'agissait d'un tableau de lignes, et Tie::File s'occupe de toutes les modifications réelles du fichier.
Par exemple il est possible de supprimer une ou des lignes avec splice(), il est possible de "retarder" l'application des modifications, de sortes que plusieurs soit appliquées à la fois.
Nous allons écrire un script qui modifie la troisième ligne d'un fichier pour lui rajouter par exemple la chaîne *", et Bob était là."*

méthode 1

1. **Méthode 1, exemple**

Avec cette méthode, on écrira ceci :

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Exemple méthode 1

```
#!/usr/bin/perl
use strict;
use warnings;

# on récupère l'argument
my $nom_fichier = shift;

# on vérifie si le fichier existe
die "Ce fichier <$nom_fichier> n'existe pas (ou n'est pas un vrai fichier)."
   if -f $nom_fichier;

# on commence par ouvrir le fichier en lecture
open my($fichier), '<', $nom_fichier
   or die "Ce fichier $nom_fichier n'a pu être ouvert : $!\n";

# on place le contenu dans un tableau de ligne
my @lignes = <$fichier>;

# on referme le fichier
close $fichier;

# on modifie la 3e ligne
chomp( $lignes[2] );
$lignes[2] .= ", et Bob était là.\n";

# on rouvre le fichier en écriture en écrasant le contenu
open my($fichier), '>', $nom_fichier
   or die "Ce fichier $nom_fichier n'a pu être ouvert : $!\n";

# et on réécrit le contenu modifié
print $fichier @lignes;

close $fichier

__END__
On a fini.
```

Il y a des variantes bien sûr, par exemple il n'est pas toujours avantageux de lire dans un tableau de lignes (quand les modifications que l'on veut faire ne sont pas basés sur une vision ligne par ligne du fichier).

méthode 2

1 Méthode 2, exemple 1

Voici une fonction qui implémente la 2ème méthode, et un exemple d'utilisation de cette fonction qui fait la même chose que l'exemple précédent :

Méthode 2, exemple 1

```
#!/usr/bin/perl
use strict;
use warnings;

use File::Copy qw(move);
use File::Temp;

# on crée une fonction qui prend un bloc et un nom de fichier en paramètre
# et exécute le bloc à chaque ligne du fichier :
# dans le bloc $_ vaut la ligne courante, et toute modification de $_
# est reflétée sur la ligne dans le fichier
sub modify_in_place (&$);

my $filename = shift;

# $. vaut le numéro de ligne dans le fichier qu'on est en train de lire
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Méthode 2, exemple 1

```
modify_in_place { s/$/, et Bob était là./ if $. == 3 } $filename;

sub modify_in_place (&$) {
    my $block    = shift;
    my $filename = shift;
    local $_;
    open my ($file), '<', $filename
      or die "This file $filename couldn't be opened : $!\n";

    # on utilise File::Temp pour créer un fichier temporaire en toute sécurité
    my $tempfile = new File::Temp();

    while (<$file>) {
        $block->();
        print $tempfile $_;
    }
    close $file;
    close $tempfile;
    move "$tempfile", $filename
      or die "We couldn't overwrite $filename with $tempfile : $!\n";
}

__END__
```

Dans ce code j'utilise File::Temp et File::Copy, ces deux modules sont distribués en standard avec Perl depuis la version 5.6.1 (5.2 pour File::Copy). Si votre version est plus vieille, vous pouvez créer un fichier temporaire à la main ou installer le module File::Temp si c'est possible. Pour ceux qui s'intéressent à des techniques "avancées" en Perl, remarquez que le prototype (&$) attribué à ma fonction m'autorise à l'utiliser comme un grep() ou un map() (bloc nu et pas de virgule). Il s'agit là d'une utilisation correcte des prototypes en Perl, ils n'ont jamais été conçu pour vérifier le type des arguments, mais plutôt pour permettre de créer des fonctions dont la syntaxe soit proche de celle des built-ins.

- **Méthode 2, exemple 2**

Voici un autre exemple qui implémente toujours la méthode 2 sans utiliser les prototypes et la syntaxe proche de celle des built-ins. Mais gardez toujours en tête que cette méthode est intéressante quand il y a plusieurs modifications à faire dans le fichier et qu'en plus, que l'on ne sache pas forcément les lignes à modifier.

Méthode 2, exemple 2

```
#!/usr/bin/perl
use strict;
use warnings;
use File::Temp;

# creation du fichier temporaire
my ($fh_temp, $file_temp)
    = tempfile( "fichier_temporaireXXXX", UNLINK => 1);

# Ouverture du fichier à modifier
open (my $fh, '<', $file) or die "Can't open $file : $!\n";

# declaration d'un compteur
my $compteur = 0;

# Lecture du fichier ligne à ligne
while(my $ligne = <$fh>) {
  $compteur ++;
  chomp ($ligne);
  # modification de la ligne
  if ($compteur == 3) {
    $ligne = $ligne.", et Bob était là.";
  }
  # ecriture dans le fichier temporaire
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Méthode 2, exemple 2

```
    print {$fh_temp} "$ligne\n";
  }
  close($fh_temp);
  close($fh);

  # copie du fichier temporaire
  rename($file_temp,$file);
```

- **méthode 3**

La méthode 3 est relativement peu intéressante, sinon à titre pédagogique, ou dans le cas d'enregistrements de taille fixe (on peut dans ce cas ouvrir le fichier en lecture-écriture et modifier directement les enregistrements sans problème de décalage puisqu'ils ne changent jamais de taille, il faut alors savoir se servir de seek() et tell()). Je ne couvrirai donc pas la méthode 3 avec un exemple, n'hésitez pas si vous avez envie d'en apporter un.

- **méthode 4**

Voici donc un exemple utilisant la méthode 4, avec Tie::File, le faire manuellement est assez inutile. L'effet sur le fichier est toujours le même que dans les exemples précédents : rajouter *", et Bob était là."* à la fin de la troisième ligne.

exemple 1 méthode 4

```
  #!/usr/bin/perl
  use strict;
  use warnings;

  use Tie::File;

  my $filename = shift;

  # @lines représente maintenant les lignes du fichier
  # $object est l'objet de classe Tie::File sous-jacent
  # sur lequel on peut appeler les méthodes de Tie::File
  my $object = tie my (@lines), 'Tie::File', $filename
    or die "We couldn't tie $filename to an array in readwrite mode : $!\n";

  $lines[2] .= ', et Bob était là.';

  # ces deux lignes (undef et untie) sont l'équivalent d'un close()
  # dans notre cas
  undef $object;
  # cette ligne suffit si on n'a pas récupéré $object (par exemple
  # dans ce script on aurait pu s'en passer)
  untie @lines;

  __END__
```

Version courte débarrassée des bouts inutiles :

Exemple 2 méthode 4

```
  #!/usr/bin/perl
  use strict;
  use warnings;

  use Tie::File;

  my $filename = shift;

  tie my (@lines), 'Tie::File', $filename
    or die "We couldn't tie $filename to an array in readwrite mode : $!\n";
  $lines[2] .= ', et Bob était là.';
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Exemple 2 méthode 4

```
__END__
```

Comme vous le voyez c'est assez élégant. Comme en plus c'est très efficace (si vous l'utilisez correctement), c'est une solution intéressante. Tie::File est distribué en standard avec Perl depuis la 5.8 mais le module en lui-même est compatible jusqu'à la 5.4.

Compatibilité Unix/Mac/Linux/Windows des fichiers (^M)

Auteurs : Stoyak , Djibril ,

- **Pour vous éviter quelques petits soucis!!**

Vous travaillez sous MAC OS X, vous développez, je peux donc supposer que vous utilisez le Terminal, qui est un système Unix!
Voilà donc un des petits soucis que j'ai rencontré, et que je voudrai vous épargner.
***Remarque** : Sachez que ces soucis de caractères ^M peuvent intervenir lorsque vous passez d'un système Windows à Linux.*
Je parsais un fichier ".txt" généré à partir d'Excel Microsoft (pour ne pas le nommer) avec un script Perl. Seulement voilà,
le système ne reconnaissait pas les lignes : le retour chariot était remplacé par un caractère ^M. Autant vous dire que mon petit script était alors totalement inefficace!! Et pourquoi?
Parce que le codage de fin de ligne diffère selon le système utilisé! Les éditeurs de texte qui ne supportent pas les retours chariots affichent ce ^M superflu!
Voilà une petite méthode pour s'en débarrasser! Ajouter cette procédure à tous vos scripts, et vous n'aurez plus de problème d'incompatibilité de fichiers Unix/Windows!

- **Procédure en question!!**

```
#!/usr/bin/perl
use warnings;
use strict;
use Carp;

# declaration des fichiers
# mon fichier à traiter
my $fichier_initial = "mon_fichier.txt";
# le fichier modifie que je cree
my $fichier_modifie = "fichier_modifie.txt";

# appel de la procedure: - en entree: mon fichier à traiter
#- en sortie: mon fichier modifie
traite_fichier($fichier_initial, $fichier_modifie);

# procedure qui permet de rendre les fichiers compatibles unix/windows
sub traite_fichier {
  # passage des parametres $fichier <- $fichier_initial
  #$fichier_modif <- $fichier_modifie
  my ($fichier, $fichier_modif) = @_;

  # lecture du fichier initial
  open ( my $FhLecture, '<', $fichier) || die ("pb d'ouverture du fichier $fichier $! ");
  # ecriture du fichier modifie
  open ( my $FhEcriture, '>', $fichier_modif) || die
 ("pb d'ecriture dans le fichier $fichier_modif $! ");
  while (my $ligne = <$FhLecture>){
    # remplacement des retours chariots!
    $ligne =~ s/\r\n?/\n/g;
    print {$FhEcriture} $case;
  }
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
    close ($FhLecture);
    close ($FhEcriture);
  }
```

- **Idem en une seule ligne de commande**

Il est possible de résoudre ce problème de compatibilité en une seule ligne de commande.

```
perl -pi.bak -e "s/\r\n/\n/" toto.txt
```

Cette ligne de commande supprimera les ^M d'un fichier et de plus, le fait d'écrire *-pi.bak* créera une sauvegarde du fichier en *toto.bak*. vous aurez donc *toto.txt* modifié et *toto.bak* équivalement à l'ancien *toto.txt*.
Pour ne pas faire de sauvegarde, il faudra écrire

```
perl -pi -e "s/\r\n/\n/" toto.txt
```

Comment transposer un fichier tabulé ?

Auteurs : Djibril ,

Si vous souhaitez transposer un fichier tabulé, c'est à dire que les lignes de votre fichier deviennent des colonnes, voici une procédure qui peut vous aider.

Entete1	Entete2	Entete3
L1Col1	L1Col2	L1Col3
L2Col1	L2Col2	L2Col3
L3Col1	L3Col2	L3Col3

Exemple de code

```
#!/usr/bin/perl
use strict;
use warnings;

TransposerFichier('tabule.txt','transpose.txt');

#=========================================
# TransposerFichier
# Transposer un fichier tabule
#=========================================
sub TransposerFichier {
  my ( $FichierTabuleOriginal, $FichierTranspose ) = @_;

  my %HashTranspose;

  # Lecture du fichier tabule
  open( my $FH, '<', $FichierTabuleOriginal )
    or die("Impossible de lire le fichier $FichierTabuleOriginal\n");

  while ( my $Line = <$FH> ) {
    chomp $Line;
    my @data = split( /\t/, $Line );
    for ( my $i = 0; $i < scalar(@data); $i++ ) {
      $HashTranspose{$i} .= $data[$i] . "\t";
    }
  }
  close($FH);

  # Création du fichier transpose
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Exemple de code

```
  open( my $FHTranpose, '>', $FichierTranspose )
    or die("Impossible de creer le fichier $FichierTranspose\n");

  foreach ( sort { $a <=> $b } keys %HashTranspose ) {
    $HashTranspose{$_} =~ s{\t$}{};
    print {$FHTranpose} "$HashTranspose{$_}\n";
  }
  close($FHTranpose);

  return $FichierTranspose;
}
```

Entete1	L1Col1	L2Col1	L3Col1
Entete2	L1Col2	L2Col2	L3Col2
Entete3	L1Col3	L2Col3	L3Col3

Comment convertir un fichier Excel en fichier csv ou txt ?

Auteurs : Djibril ,

Le code ci-dessous permet de convertir un fichier excel 2007 (ou antérieur) en fichier txt, csv ou autre fichier plat.
Le choix du séparateur est laissé à l'utilisateur.
Si l'on on précise un répertoire, le fichier convertit sera créé dans ce dernier, sinon, il sera créé dans le même répertoire que le fichier excel.
On peut choisir de convertir toutes les feuilles ou non du fichier excel en mettant -feuilles à 1 => tout sera convertit, ou à 0 et dans ce cas, à chaque feuille une confirmation sera demandée.

Conversion d'un ficher excel en un fichier texte

```
#!/usr/bin/perl
use warnings;
use strict;

my %Arguments = (
  -fichier    => "/chemin/vers/fichier.xls",    # notre fichier excel
  -type       => "csv",                         # ou txt, ou autre
  -separateur => ";",                           # ou "\t" , "|", ... au choix
  -repertoire => "/autre/repertoire",           # par defaut, repertoire fichier.xls
  -feuilles   => 1,                             # ou 0 => demande confirmation
);
my $fichier = ExcelToCsvTxt( \%Arguments );

#=============================================================================
# But       : Convertit un fichier excel en txt, csv ou autre fichier plat
# Arguement : Une référence de hash
# Retourne  : fichier txt ou csv
# Necessite : Spreadsheet::ParseExcel, Spreadsheet::XLSX et File::Basename
#=============================================================================
sub ExcelToCsvTxt {
  my $RefArgument = shift;

  my $FichierXls = $RefArgument->{-fichier};
  my $Type       = $RefArgument->{-type};
  my $Repertoire = $RefArgument->{-repertoire};
  my $Separateur = $RefArgument->{-separateur};
  my $feuilles   = $RefArgument->{-feuilles};
  $feuilles = 1 unless defined $feuilles;

  # vérification du fichier
  unless ( defined $FichierXls and $FichierXls =~ /\.xlsx?$/i ) {
    die <<'USAGE';
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Conversion d'un ficher excel en un fichier texte

```
        my %Arguments = (
          -fichier    => "/chemin/vers/fichier.xls",    # notre fichier excel
          -type       => "csv",                         # ou txt, ou autre
          -separateur => ";",                           # ou "\t" , "|", ... au choix
          -repertoire => "/autre/repertoire",           # par defaut, repertoire fichier.xls
          -feuilles   => 1,                             # ou 0 => demande confirmation
        );
        my $fichier = ExcelToCsvTxt( \%Arguments );

USAGE
  }

  # vérification du type de conversion voulu
  unless ( defined $Type and $Type =~ m{^csv|txt$}i ) {
    $Type = 'txt';
  }

  require File::Basename;

  # vérification du type de conversion voulu
  unless ( defined $Repertoire and -d $Repertoire ) {
    $Repertoire = File::Basename::dirname($FichierXls);
  }

  # Nouveau fichier
  my (@FileParse) = File::Basename::fileparse( $FichierXls, qr/\.[^.]*/ );
  my $FichierTxtCsv = $Repertoire . '/' . $FileParse[0] . ".$Type";

  # On verifie si c'est un fichier excel version 2007 ou plus ancien
  my $ExcelObj;
  if ( $FichierXls =~ m{\.xls$}i ) {
    require Spreadsheet::ParseExcel;
    $ExcelObj = Spreadsheet::ParseExcel::Workbook->Parse($FichierXls)
      or die("Impossible de lire le fichier $FichierXls\n");
  }
  else {
    require Spreadsheet::XLSX;
    $ExcelObj = Spreadsheet::XLSX->new($FichierXls)
      or die("Impossible de lire le fichier $FichierXls\n");
  }

  # Nombre de feuilles dans le fichier excel
  my $NbrFeuilles = scalar @{ $ExcelObj->{Worksheet} };

  # Création du fichier final
  open( my $fh, '>', $FichierTxtCsv )
    or die "impossible de ceer le fichier $FichierTxtCsv\n";

  my ( $iR, $iC, $ObjetFeuille, $oWkC );

  foreach my $ObjetFeuille ( @{ $ExcelObj->{Worksheet} } ) {

    if ( $NbrFeuilles > 1 and $feuilles != 1 ) {
      print "Voulez vous convertir l'onglet '$ObjetFeuille->{Name}' [Y/n] : ";
      chomp( my $Reponse = <STDIN> );
      print "\n";
      next if ( defined $Reponse and uc($Reponse) eq 'N' );
    }

    # Parcours des lignes
    for (
      my $iR = $ObjetFeuille->{MinRow};
      defined $ObjetFeuille->{MaxRow} && $iR <= $ObjetFeuille->{MaxRow};
      $iR++
      )
    {
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Conversion d'un ficher excel en un fichier texte

```
        # Parcours des colonnes
        for (
          my $iC = $ObjetFeuille->{MinCol};
          defined $ObjetFeuille->{MaxCol} && $iC <= $ObjetFeuille->{MaxCol};
          $iC++
          )
        {
          $oWkC = $ObjetFeuille->{Cells}[$iR][$iC];
          if ( defined $oWkC ) {
            print {$fh} $oWkC->Value, $Separateur;
          }
          else {
            print {$fh} $Separateur;
          }
        }
        print {$fh} "\n";
      }
    }
    close($fh);

    return $FichierTxtCsv;
  }
```

Attention: : Nous avons utilisé ici la méthode Value pour récupérer la valeur de la case

```
  print {$fh} $oWkC->Value, $Separateur;
```

Mais cette méthode peut nous générer des erreurs inattendues :

explication documentation officielle

```
value()

The value() method returns the formatted value of the cell.

    my $value = $cell->value();

Formatted in this sense refers to the numeric format of the cell value.
For example a number such as 40177 might be formatted as 40,117, 40117.000 or even as the date
 2009/12/30.

If the cell doesn't contain a numeric format then the formatted and unformatted
cell values are the same, see the unformatted() method below.

For a defined $cell the value() method will always return a value.

In the case of a cell with formatting but no numeric or string contents
the method will return the empty string ''.
```

Pour éviter tout souci, utilisez plutôt la méthode unformatted.

```
  print {$fh} $oWkC->unformatted, $Separateur;
```

Comment fusionner plusieurs classeurs Excel d'un répertoire en un unique fichier ?

Auteurs : Djibril ,

Ce code vous permettra de fusionner plusieurs classeur Excel (xls, xlsx) en un unique fichier Excel (xls). Il copie toutes les feuilles des différents classeurs dans un seul fichier. Le seul inconvénient est qu'il ne garde pas les noms des feuilles mais en génère automatiquement (feuille1, 2, 3, ...) et les formats des cellules ne sont pas conservés.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Vous aurez besoin d'installer les modules

1. Spreadsheet::ParseExcel
2. Spreadsheet::XLSX
3. Spreadsheet::WriteExcel

Les modules Getopt::Long et Pod::Usage sont dans le core de Perl.

Pour savoir comment lancer le programme, faites

```
perl nom_programme.pl -help
```

ou

```
perl nom_programme.pl -man
```

```
#!/usr/bin/perl
#===============================================================================
# Author :  Djibril
# Date   :  23/02/2011 10:24:10
# Main   :  Fusionner plusieurs classeurs Excel d'un répertoire en un unique fichier
#===============================================================================
use Carp;
use strict;
use warnings;

use Spreadsheet::ParseExcel;
use Spreadsheet::XLSX;
use Spreadsheet::WriteExcel;
use Getopt::Long;
use Pod::Usage;

my ( $repertoire_excel, $fichier_excel_fusionne ) = ();
my ( $man, $help ) = ( 0, 0 );

GetOptions(
  'repertoire|d=s' => \$repertoire_excel,
  'output|o=s'     => \$fichier_excel_fusionne,
  'help|?'         => \$help,
  'man'            => \$man
) or pod2usage(2);

pod2usage( -exitstatus => 0, -verbose => 2 ) if ( $man || $help );
if ( ( !defined $repertoire_excel or !defined $fichier_excel_fusionne ) ) {
  pod2usage( { -verbose => 1, -output => \*STDERR } );
}

# Lister les fichiers excel d'un répertoire et de ses sous répertoires
my @fichiers_excel = lister_fichiers($repertoire_excel);

fusionner_fichiers_excel( \@fichiers_excel, $fichier_excel_fusionne );

# Procédure de fusion excel
sub fusionner_fichiers_excel {
  my ( $ref_fichiers_excel, $fichier_excel_final ) = @_;

  print "Creation de $fichier_excel_fusionne\n";

  # Liste des fichiers excel à fusionner
  my @les_fichiers_excel = @{$ref_fichiers_excel};
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
  # Creation du classeur Excel
  my $workbook_final = Spreadsheet::WriteExcel->new($fichier_excel_final);

  foreach my $fichier_excel (@les_fichiers_excel) {
    next unless ( $fichier_excel =~ m{\.xlsx?$}i );

    # Lecture du fichier Excel
    print "Lecture du fichier $fichier_excel\n";
    my $workbook;
    if ( $fichier_excel =~ m{\.xls$} ) {
      my $parser = Spreadsheet::ParseExcel->new();
      $workbook = $parser->parse($fichier_excel);
    }
    else {
      $workbook = Spreadsheet::XLSX->new($fichier_excel);
    }

    if ( !defined $workbook ) { die "Erreur de lecture du fichier $fichier_excel : "; }

    for my $worksheet ( $workbook->worksheets() ) {
      my ( $row_min, $row_max ) = $worksheet->row_range();
      my ( $col_min, $col_max ) = $worksheet->col_range();
      next if ( $row_max == 0 and $col_max == 0 );

      # Création de la feuille
      my $worksheet_final = $workbook_final->add_worksheet();

      for my $row ( $row_min .. $row_max ) {
        for my $col ( $col_min .. $col_max ) {

          my $cell = $worksheet->get_cell( $row, $col );
          next unless $cell;
          $worksheet_final->write( $row, $col, $cell->unformatted() );
        }
      }
    }
    print "\n";

  }
  $workbook_final->close();

  return;
}

#=========================================================
# Nombre d'arguments : 1
# Argument(s)        : un répertoire ($repertoire)
# Retourne           : Tableau de fichier (@fichiers)
#=========================================================
sub lister_fichiers {
  my ($repertoire) = @_;
  my @fichiers;

  # Ouverture d'un répertoire
  opendir( my $fh_rep, $repertoire )
    or die "impossible d'ouvrir le répertoire $repertoire\n";

  # Liste fichiers et répertoire sauf (. et ..)
  my @Contenu = grep { !/^\.\.?$/ } readdir($fh_rep);

  # Fermeture du répertoire
  closedir($fh_rep);

  # On récupère tous les fichiers
  foreach my $nom (@Contenu) {

    # Fichiers
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
    if ( -f "$repertoire/$nom" ) {
      push( @fichiers, "$repertoire/$nom" );
    }

    # Repertoires
    elsif ( -d "$repertoire/$nom" ) {

      # recursivité
      push( @fichiers, lister_fichiers("$repertoire/$nom") );
    }
  }

  return @fichiers;
}

__END__

=head1 NAME

    Ce programme permet de fusionner plusieurs classeurs excel (xls et xslx) d'un repertoire en un fichier Excel

    Il copie toutes les feuilles des differents classeurs dans un seul fichier. Le seul inconvenient est qu'il n
    pas les noms des feuilles mais en genere automatiquement (feuille1, 2, 3, ...) et les formats des cellules n
    pas conserves, uniquement le contenu des cellules.

=head1 SYNOPSIS

  perl programme_perl.pl -d "C:/REPERTOIRE/EXCEL" -o "fichier_excel_fusionne.xls"
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Comment connaitre le nom et l'ip d'une machine en perl ?

Auteurs : Djibril ,

Pas de secret, le CPAN contient ce qu'il faut. Utilisez les modules Sys::Hostname et Socket déjà présent dans le CORE de perl.

IP
```perl
#!/usr/bin/perl
use strict;
use warnings;

use Sys::Hostname;
use Socket;

my $host = hostname;
my $IP = inet_ntoa(inet_aton(hostname()));

print "Ma machine $host a pour adresse IP : $IP\n";
```

Comment connaitre le pid du script perl en cours ?

Auteurs : Djibril ,

Pour connaitre le pid du script perl en cours, il suffit d'utiliser la variable spéciale perl $$. Ainsi, faite ceci :

pid
```perl
#!/usr/bin/perl
use strict;
use warnings;

print "mon pid : $$\n";
```

Comment connaitre le système d'exploitation de ma machine ?

Auteurs : Djibril ,

Vous pouvez soit afficher une variable spéciale perl $^O ou bien afficher le contenu de la variable d'environnement %ENV. Voici un exemple de script affichant les deux façon d'obtenir le résultat voulu.

Variable spéciale - préconisé
```perl
#!/usr/bin/perl
use strict;
use warnings;
my $sys_expl = $^O;
print "Voici mon OS : $sys_expl\n"; # => Voici mon OS : MSwin32
```

Resultat sur 3 type d'OS
```
sous Windows : mMSwin32
  sous Linux   : linux
  sous MAC     : darwin
```

ou

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Avec la variable d'environnement

```
#!/usr/bin/perl
use strict;
use warnings;

#Pour information
foreach my $nom_cle (keys %ENV) {
  print "$nom_cle : $ENV{$nom_cle}\n";
}

# En ce qui nous interesse
print "Voici mon OS : $ENV{OS}\n";  # Voici mon OS : Windows_NT
```

Vous pouvez constatez que le résultat nous indique bien qu'on est sous windows. Mais les deux sont légèrement différents. Néanmoins, $^O donnera toujours le même résultat.

Comment déterminer l'espace des disques ?

Auteurs : Djibril ,

Pour connaitre l'espace disque restant du C:/, D:/ ou sous linux du /, ... Il existe un module Perl, Filesys::DfPortable.

```
#!/usr/bin/perl
use strict;
use warnings;

use Filesys::DfPortable;

my $Disque = 'C:/'; # ou même / par exemple

my $ref = dfportable($Disque);
if( defined($ref) ) {
 print"Total bytes: $ref->{blocks}\n";
 print"Total bytes free: $ref->{bfree}\n";
 print"Total bytes avail to me: $ref->{bavail}\n";
 print"Total bytes used: $ref->{bused}\n";
 print"Percent full: $ref->{per}\n"
}
```

Cet exemple vous donnera la taille en octets. Si vous souhatez la taille en ko, Mo ou plus, il vous suffit de fournir un deuxième arguement.

```
#!/usr/bin/perl
use strict;
use warnings;

use Filesys::DfPortable;

my $Disque = 'C:/'; # ou même / par exemple
my $FormatSortie = 1024*1024*1024;
my $ref = dfportable($Disque, $FormatSortie);
if( defined($ref) ) {
 print"Total bytes: $ref->{blocks}\n";
 print"Total bytes free: $ref->{bfree}\n";
 print"Total bytes avail to me: $ref->{bavail}\n";
 print"Total bytes used: $ref->{bused}\n";
 print"Percent full: $ref->{per}\n"
}
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
# Le résultat sera en Go.
```

Comment lancer des commandes sur un serveur distant depuis Windows ?

Auteurs : Djibril ,

Il peut être intéressant de se connecter sur un serveur distant Linux afin de lancer des commandes telles (ls, perl toto.pl, etc) depuis un PC Windows. De plus, l'idéal serait de transférer des fichiers depuis son PC Windows vers un autre serveur ou l'inverse. Il est possible de faire cela depuis un script Perl sans avoir besoin d'avoir cygwin, putty, ...

Voici une procédure utilisant le module Net::SSH2.

Faire du SSH depuis Windows

```
#=====================================================
# But  : Connection SSH + commandes (scp ou autre)
# Args : Reference d'un hash
# Retourne : rien
# Besoin   : modules Net::SSH2
#=========================
sub CommandSSHFromWindows {
    unless ( scalar(@_) == 1 ) {
        my $usage = <<'FIN_USAGE';
     Usage:
     my %DataSSH = (
       host     => "IP serveur ou Host serveur",
       login    => "login",
       password => "xxxxxx",
       cmd      => ["perl /home/toto/test.pl", "ls /usr"],
       scp_put  => [
         [ "Fichier local", "Fichier destination"],
         [ "Fichier local"], # Depot dans le rep courant du user
       ],
       scp_get  => [
         [ "Fichier destination", "Fichier local"],
         [ "Fichier destination"], # Depot dans le rep local courant
       ],
       verbose  => 1, # ou 0
     );
     CommandSSHFromWindows( \%DataSSH);

FIN_USAGE
        die($usage);
    }
    my ($RefDataConnection) = @_;
    $|++;

    # parameters
    my $host     = $RefDataConnection->{host};
    my $login    = $RefDataConnection->{login};
    my $password = $RefDataConnection->{password};
    my $Refcmd   = $RefDataConnection->{cmd};
    my $Verbose  = $RefDataConnection->{verbose} || 0;

    require File::Basename;
    require Net::SSH2;

    my $ssh2 = Net::SSH2->new();
    $ssh2->connect($host);

    my ( $code, $error_name, $error_string ) = $ssh2->error();
    if ($code) {
        print "[WARNING] Unable to connect to host : $host\n\n";
        return;
    }
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Faire du SSH depuis Windows

```
    unless ( $ssh2->auth_password( $login, $password ) ) {
        $ssh2->disconnect();
        print "[WARNING] Unable to login.\n"
            . "Check your login ($login) or your password\n\n";
        return;
    }

    foreach my $cmd ( @{$Refcmd} ) {
        if ( $Verbose == 1 ) {
            print "\n- $cmd\n";
        }
        my $channel = $ssh2->channel();
        $channel->blocking(1);
        $channel->exec($cmd);

        while ( $channel->read( my $buffer, 1024 ) ) {
            if ( $Verbose == 1 ) {
                print $buffer;
            }
        }
        $channel->close;
    }

    # SCP si necessaire
    # Put
    if ( exists $RefDataConnection->{scp_put} ) {
        foreach my $RefCoupleSCP_put ( @{ $RefDataConnection->{scp_put} } ) {
            unless ( -e $RefCoupleSCP_put->[0] ) {
                print "\n$RefCoupleSCP_put->[0] n'existe pas \n";
                next;
            }
            my $message = "Transfert de $RefCoupleSCP_put->[0] vers $host";
            if ( $Verbose == 1 ) {
                print "\n$message ... ";
            }

            $ssh2->scp_put( $RefCoupleSCP_put->[0], $RefCoupleSCP_put->[1] )
                or die $ssh2->error();

            print "fini\n" if ( $Verbose == 1 );
        }
    }

    # Get
    if ( exists $RefDataConnection->{scp_get} ) {
        foreach my $RefCoupleSCP_get ( @{ $RefDataConnection->{scp_get} } ) {
            my $NomFichier = File::Basename::basename( $RefCoupleSCP_get->[0] );
            if ( $Verbose == 1 ) {

            }

            $ssh2->scp_get( $RefCoupleSCP_get->[0], $RefCoupleSCP_get->[1] )
                or die $ssh2->error();

            print " : fini\n" if ( $Verbose == 1 );
        }
    }

    $ssh2->disconnect();
    return;
}
```

Pour l'utiliser, Il faut donner en argument une référence de hachage. Si vous souhaitez afficher les messages affichés sur la console du serveur distant, mettez verbose à 1.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
#!/usr/bin/perl
use warnings;
use strict;
use Carp;

my %DataSSH = (
    host     => "Votre host",
    login    => "mon login",
    password => "mon mot de passe",
    cmd      => [ "ls -alh /home/user", "perl toto.pl" ],
    verbose => 1,    # ou 0
);
CommandSSHFromWindows( \%DataSSH );
```

Si vous souhaitez envoyer un ou plusieurs fichiers sur le serveur distant, rajoutez dans le hachage la clé scp_put et/ou scp_get, exemple

```
scp_put  => [
          # Fichier local       # fichier distant
    [ "C:/Rep/path/monfichier sur mon disque", "/home/user/toto"],
    [ "C:/Rep/path/monfichier2 sur mon disque"],
    [ "C:/Rep/path/monfichier3 sur mon disque", "/home/user/monfichier3"],
    #...
   ],
```

Voilà, si vous avez des questions, des remarques ou suggestions, c'est ici.

Comment se connecter via SSH sur des périphériques CISCO ?

Auteurs : michon ,

Ayant eu le problème moi même et constatant que c'est un problème récurrent sur Internet dont on ne trouve pas de vrais solutions, je vous propose la mienne. Pour SSH sur les périphériques Cisco beaucoup utilisent les modules Net::SSH::Perl, Perl::SSH2, voir même Net::Appliance::session et aboutissent à des problèmes sur les périphériques Cisco du genre :

- **Connexion SSH impossible**
- **Passage Enable impossible**
- **Impossible de lancer plusieurs commandes dans la même session ...**

Donc après avoir tout essayé, je suis tombé sur le module Net::SSH::Expect qui m'a permis d'obtenir une solution fonctionnelle en s'affranchissant des problèmes liés aux autres modules.

Exemple de code

```
#!/usr/bin/perl
use strict;
use warnings;

use Net::SSH::Expect;

my $ssh = Net::SSH::Expect->new(
  host     => '192.168.90.100',
  password => 'cisco',
  user     => 'admin',
  raw_pty  => 1
);

my $enable_passwd = "cisco";
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Exemple de code

```
my $login_output  = $ssh->login();

$ssh->send("enable");
$ssh->waitfor( 'Password:\s*\z', 1 ) or die "prompt 'password' not found after 1 second";
$ssh->send($enable_passwd);

my $ls = $ssh->exec("show vlan");
print "$ls\n";

#Première façon de récupérer une sortie longue:
$ssh->send("show running-config");
while ( my $line = $ssh->read_line() ) {
  print "$line\n";
}

#Deuxième façon (mais on récupère aussi les prompt avec cette méthode):
my @output = $ssh->exec("show running-config");
print @output;
$ssh->close();
```

Pour les problèmes de longueur de sortie sur le terminal (passage du --more-- sur les Cisco), il suffit d'exécuter la commande suivante avant votre commande :

```
$ssh->exec("terminal length 0");
```

et celle ci après votre commande afin de revenir dans l'état par défaut :

```
$ssh->exec("terminal length 24");
```

Bien sûr, les commandes "show vlan" et/ou "show running-config" sont là à titre indicatif, à remplacer par vos propres commandes ...

Voilà, si vous avez des questions, des remarques ou suggestions, c'est ici.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > Codes sources utiles > Des codes sources > Gestions des dates

Comment afficher une date en perl ?

Auteurs : Djibril , Philou67430 ,

Il existe plusieurs fonctions perl permettant d'obtenir la date (année, mois, jour, heure, etc.).

1. **time : renvoie un nombre scalaire qui indique le nombre de secondes qui se sont écoulées depuis la date fr epoch de votre machine.**
2. **localtime : renvoie les données relatives au temps local. L'affichage ressemble à celui de la fonction date de linux.**
3. **gmtime : renvoie les données relatives au temps universel.**
4. **Module POSIX.**

Affichage des dates

```perl
#!/usr/bin/perl
use strict;
use warnings;
my $le_time = time;
print "Depuis le 1/1/1970, il y a eu $le_time secondes.\n";
my $le_localtime = localtime;
print "La date locale : $le_localtime\n";
my $le_gmtime = gmtime;
print "La date universelle : $le_gmtime\n";
```

Résultat

```
Depuis le 1/1/1970, il y a eu 1185539208 secondes.
La date locale : Fri Jul 27 14:26:48 2007
La date universelle : Fri Jul 27 12:26:48 2007
```

localtime(time) renvoie un tableau contenant plusieurs données nécessaires comme le jour, le mois, l'année, etc. Il est plus facile d'expliquer via un exemple. Voici une procédure que j'utilise régulièrement pour afficher la date.

Affichage des dates 2

```perl
#!/usr/bin/perl
use strict;
use warnings;

my $RefDateActuelle = date(); # On recupere la réference d'un hash
my $time = 1234567890;
my $RefAutreDate    = date($time); # On recupere la réference d'un hash


print "Date actuelle : $RefDateActuelle->{date} $RefDateActuelle->{heure}\n";
print "Date correspondant au time $time : $RefAutreDate>{date} $RefAutreDate>{heure}\n";

sub date {
  my $time = shift || time;    #$time par defaut vaut le time actuel
  my
 ( $seconde, $minute, $heure, $jour, $mois, $annee, $jour_semaine, $jour_annee, $heure_hiver_ou_ete
 )
    = localtime($time);
  $mois  += 1;
  $annee += 1900;

  # On rajoute 0 si le chiffre est compris entre 1 et 9
  foreach ( $seconde, $minute, $heure, $jour, $mois, $annee ) {
    s/^(\d)$/0$1/;
  }

  my %date = (
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Affichage des dates 2
```
      "date"         => "$jour-$mois-$annee",
      "heure"        => "$heure:$minute:$seconde",
      "jour_semaine" => $jour_semaine,
      "jour_annee"   => $jour_annee,
      "hiverOuEte"   => $heure_hiver_ou_ete,
    );
    return \%date;
  }
```

```
Date actuelle : 05-12-2008 11:02:57
Date correspondant au time 1234567890 : 14-02-2009 00:31:30
```

POSIX

exemple 1
```
use POSIX qw(strftime);
my $now_string = strftime "%a %b %e %H:%M:%S %Y", localtime;
```

Exemple 2 en utilisant le format %V, qui défini le numéro de semaine selon la norme ISO 8601:2000

exemple 2
```
use POSIX qw(strftime);
my %jours = ("1er janvier 2009" => [ 0, 0, 0, 1, 0, 109 ],
             "31 décembre 2009" => [ 0, 0, 0, 31, 11, 109 ]);
print strftime "Le $_ est en semaine %V\n", @{$jours{$_}} foreach sort keys %jours;
```

Qui produit :

résultat exemple 2
```
Le 1er janvier 2009 est en semaine 01
Le 31 décembre 2009 est en semaine 53
```

Comment récupérer une date aléatoire entre deux dates données ?

Auteurs : Philou67430 ,

Comme DateTime est le module préconisé, mais qu'il n'est pas disponible dans le Core, alors que POSIX l'est, il me semble que c'est une bonne alternative pour des utilisations "limitées".
Le code ci-dessous est compatible avec des machines acceptant des entiers sur 32 bits. Il utilise mktime et localtime.
Attention toutefois sur les machines 32 bits, les dates limites utilisables sont de 1904 à 2038. Pour utiliser des dates en dehors de cette plage, il faut s'en remettre aux modules spécialisés comme DateTime.

```
my $min_date = mktime(0,0,0, 1, 0, 10); # 1 janvier 1910
my $max_date = mktime(0, 0, 0, 1, 0, 130); # 1 janvier 2030
my ($sec, $min, $hour, $day, $month, $year) = localtime(int(rand($max_date)-$min_date))+$min_date);
```

Comment connaitre le time depuis une date (L'inverse de localtime et gmtime) ?

Auteurs : Djibril ,

Il peut être utile de connaître le time d'une date, c'est l'inverse de time, localtime et gmtime. Cela se fait par exemple en php pour les connaisseurs via la fonction mktime. Comme d'habitude, il existe un module Time::Local en perl déjà présent dans le CORE, donc pas besoin d'installation.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Ce module posséde deux méthodes timelocal et timegm qui renvoient le time. Il faut leur donner 6 informations sous forme de tableau ($sec,$min,$hour,$mday,$mon,$year).
Attention : $year sera bien l'année classique (avec + 1900), ex : 2007 et $mois sera compris entre 0 et 11 et non entre 1 et 12.
Nous trouver le time des dates 01/01/2000 et 05/12/2008

```
#!/usr/bin/perl
use strict;
use Carp;
use warnings;

use Time::Local;

my @Dates1 = split("/", "01/01/2000" );
my @Dates2 = split("/", "05/12/2008" );

# Reduisons le mois de 1
$Dates1[1] -= 1;
$Dates2[1] -= 1;

print "01/01/2000 : ", timelocal( 0,0,0, @Dates1 ),"\n";
print "05/12/2008 : ", timelocal( 0,0,0, @Dates2 ),"\n";
```

Vous pourrez maintenant jongler avec les time afin de faire des calculs sur les dates, faires des comparaisons de dates et autres. D'un point pédagogique c'est bien, mais je vous conseille d'utiliser des modules adéquats pour faire cela plus proprement.

Résultat

```
01/01/2000 : 946681200
15/12/2008 : 1229295600
```

Comment calculer le nombre de jours et/ou semaines entre deux dates ?

Auteurs : Djibril ,

Vous souhaitez connaitre le nombre de jours (ou le nombre de semaines) entre deux dates, utilisez le module Date::Calc.

```
#!/usr/bin/perl
use warnings;
use strict;
use Date::Calc qw(:all);

my $date1 = '28/05/2006';
my $date2 = '19/11/2009';
my ( $jour1, $mois1, $annee1 ) = split( '/', $date1 );
my ( $jour2, $mois2, $annee2 ) = split( '/', $date2 );

# Calcul le nombre de jour entre 2 dates
my $NombreJoursEntreDate = Delta_Days( ( $annee1, $mois1, $jour1 ), ( $annee2, $mois2, $jour2 ) );
print "Il y a $NombreJoursEntreDate jours entre $date1 et $date2\n";

# ou Nombre de semaines et jours
my $NbrSemaine     = int( $NombreJoursEntreDate / 7 );
my $NbrJourRestant = $NombreJoursEntreDate % 7;
print "Il y a $NbrSemaine semaine(s) et $NbrJourRestant jour(s) entre $date1 et $date2\n";
```

Résultat

```
Il y a 1271 jours entre 28/05/2006 et 19/11/2009
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Résultat

```
Il y a 181 semaine(s) et 4 jour(s) entre 28/05/2006 et 19/11/2009
```

Connaitre une date vieille ou futur

Auteurs : Djibril ,

Exemple de script permettant de connaitre :

1. la date d'aujourd'hui
2. la date dans 6 jours
3. la date 48 jours avant aujourd'hui
4. la date 2 ans et 6 mois avant aujourd'hui
5. la date de demain

```
#!/usr/bin/perl
use strict;
use warnings;

use Date::Calc qw(:all);
my $usage = <<"FIN_USAGE";
  Usage: perl $0 date separateur (Ex : perl $0 12/05/2003 / )
       ou perl $0
FIN_USAGE

my ($jour,$mois,$annee) = ();
if ( @ARGV == 0 ) {
  ($jour,$mois,$annee)= format_date();
}
elsif ( @ARGV == 2 ) {
   my ($date, $separateur ) = @ARGV;
   chomp $date;
   chomp $separateur;
  ($jour,$mois,$annee) = format_date($date,$separateur);
}
else {
  print "$usage\n";
  exit;
}

##############################
print "Date : $jour/$mois/$annee\n";
my ($aa,$mm,$jj,$aaa,$mmm,$jjj) = ();
($aa,$mm,$jj)= Add_Delta_Days($annee,$mois,$jour,6);
print "6 jours apres : $jj/$mm/$aa\n";

($aa,$mm,$jj)= Add_Delta_Days($annee,$mois,$jour,-48);
print "48 jours avant : $jj/$mm/$aa\n";
($aa,$mm,$jj)= Add_Delta_YM($annee,$mois,$jour,-2,-6);
print "2 ans et 6 mois avant : $jj/$mm/$aa\n";

($aaa,$mmm,$jjj)= Add_Delta_Days($annee,$mois,$jour,1);
print "jour d'apres : $jjj/$mmm/$aaa\n";

sub format_date {
  my ($date, $separateur) = @_;
  my ($jour,$mois,$annee) = ();
  if (defined $date) {
    ($jour,$mois,$annee) = split($separateur,$date);
    return ($jour,$mois,$annee);
  }
  else {
    ($jour,$mois,$annee) =(localtime)[3,4,5];
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
    $mois = $mois + 1;
  $annee = $annee + 1900;
    # On rajoute 0 si le chiffre est compris entre 1 et 9
    foreach  ($jour,$mois,$annee) {
      s/^(\d)$/0$1/;
    }
    return ($jour, $mois, $annee);
  }
}
```

Comment obtenir la date au format DB2, obtenir la microseconde ?

Auteurs : Djibril , didleur ,

Voici une procédure pour générer un timestamp au format DB2 c'est à dire au format : AAAA-MM-JJ HH:mm:ss.xxxxxx où les x représentent les microsecondes.

```
#!/usr/bin/perl
use strict;
use warnings;
use Time::HiRes;

print ObtenirDateFormatDB2(),"\n";
# ------------------------------------------------
# ObtenirDateFormatDB2
# But : Retourne un temps au foramt DB2
# ex : 2009-10-25 16:35:53.123456
# ------------------------------------------------
sub ObtenirDateFormatDB2 {
  my ( $epochsecondes, $microsecondes ) = Time::HiRes::gettimeofday;
  my ( $seconde, $minute, $heure, $jour, $mois, $annee ) = localtime($epochsecondes);

  $mois  += 1;
  $annee += 1900;

  # On rajoute 0 si le nombre est compris entre 1 et 9
  foreach  ( $seconde, $minute, $heure, $jour, $mois, $annee, $microsecondes ) {
    s{^(\d)$}{0$1};
  }

  return "$annee-$mois-$jour $heure:$minute:$seconde.$microsecondes";
}
```

Résultat : ***2009-10-23 09:37:52.957328***

N'hésitez pas à consulter la documentation du module Time::HiRes.

Les modules les plus utilisés pour la gestion des dates en perl

Auteurs : Jedai ,

Les modules les plus utilisés en perl pour la gestion des dates sont DateTime, Date::Manip, Date::Calc et POSIX.

DateTime a été fondé afin de rassembler l'ensemble des fonctionnalités en rapport avec le temps auparavant dispersées sur un grand nombre de module et d'offrir une interface simple et cohérente à ces fonctionnalités, le tout en gardant des bonnes performances et un poids très raisonnable. D'après tout ce que j'ai pu essayer avec, c'est un succès ! Je recommande donc à tous ceux qui veulent manipuler de façon complexe des dates en Perl de se diriger vers cette distribution, ils sont garantis d'y trouver ce dont ils ont besoin, et de pouvoir travailler simplement et efficacement

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

avec. Si une grande majorité de Perliste se met à utiliser DateTime, cela facilitera également les échanges de code entre utilisateurs du forum. Il est également possible que DateTime se retrouve intégré au CORE, ce qui faciliterait encore son utilisation et sa diffusion.
Vous êtes bien sûr libre d'utiliser n'importe quel module de gestion de dates (après tout la philosophie de Perl c'est TIMTOWDI : There's more than one way to do it).

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > Codes sources utiles > Des codes sources > Gestions des tableaux (Array)

Comment supprimer les doublons d'un tableau ?

Auteurs : Djibril ,

Il existe différentes méthodes pour supprimer les doublons dans une liste (un tableau) en perl. La première méthode consiste à utiliser un module Perl. C'est bien évidemment la méthode la plus recommandée car les modules disponibles sur le CPAN sont efficaces et bien écrits.
Voici une liste de modules intéressants :

- List::MoreUtils
- List::Util
- List::Uniq

La seconde méthode revient à ne pas utiliser de modules et coder soit même. Voici 3 procédures élégantes qui permettent de supprimer les doublons d'un tableau. Comparons les.

1 Technique 1 (hash anonyme)

Cette procédure prend en argument une référence d'un tableau et chaque case du tableau devient une clé d'un hash anonyme. Les clés de la table de hachage sont renvoyées.

méthode 1
```perl
#!/usr/bin/perl
use strict;
use warnings;

sub doublons_hash_anonyme {
  my ($ref_tabeau) = @_;

  return keys %{ { map { $_ => 1 } @{$ref_tabeau} } };
}
```

1 Technique 2 (tranche)

Cette procédure prend en argument une référence d'un tableau et chaque case du tableau devient la clé d'une tranche de hachage d'un hachage qu'on aura déclaré.

méthode 2
```perl
#!/usr/bin/perl
use strict;
use warnings;

sub doublons_tranche {
  my ($ref_tabeau) = @_;

  my %hash_sans_doublon;    # Comme un hash ne peut pas avoir deux clés identiques,
                            # utilser ces clé permet d'avoir un tableau unique
  @hash_sans_doublon{ @{$ref_tabeau} } = ();
  # Pas besoin de surcharger le hash avec des valeurs inutiles
                                          # et ensuite, on renvoie le tableau des clés uniques
  return keys %hash_sans_doublon;
}
```

1 Technique 3 (grep)

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Cette procédure prend en argument une référence d'un tableau et chaque case est passée au filtre grep qui ne conserve que celles qui n'ont pas déjà été vues.

méthode 3
```
#!/usr/bin/perl
use strict;
use warnings;

sub doublons_grep {
  my ($ref_tabeau) = @_;

  my %hash_sans_doublon;

  return grep { !$hash_sans_doublon{$_}++ } @{$ref_tabeau};
}
```

- **Avantages et inconvénients**

Ces trois techniques sont simples, efficaces et facilement maintenables. Il n'est pas du tout nécessaire d'installer un nouveau module pour cela, bien que *List::Util* soit dans le CORE de perl depuis sa version 5.8.
Pour de plus amples manipulations de tableaux, je vois recommande biensûr les modules cités ci-dessus.
Sachez maintenant que parmi ces 3 procédures, certaines sont plus rapides que d'autres. la technique 2 (à coup de tranche) est la plus rapide des 3, ensuite vient la technique 3 (à coup de grep), puis vient la dernière (technique 1). Ce n'est qu'une remarque, mais tout dépend de la longueur des listes, du nombre de doublons qu'elles contiennent, etc ...
En général, *la première place est toujours partagée entre la technique 2 (à coup de tranche) et 3 (à coup de grep)*.
Autres remarques :
- La technique 2 (tranche) est plus avantageuse que la 1 car elle n'a pas besoin de créer une liste intermédiaire.
- Notez bien que seule la technique 3 (grep) permet de conserver l'ordre des éléments du tableau après suppression des doublons (important selon vos besoins) ! Sachez que la différence de temps entre la 2 et 3 n'est pas non plus dramatique. Utilisez le module *Benchmark* pour effectuer vos tests si vous le souhaitez.

1 Exemple d'utilisation

Exemple
```
#!/usr/bin/perl
use strict;
use warnings;

my @listing_mots = qw/ moi 2Eurocents Woufeil toi Djibril Jedai moi Jedai gldavid
  etc Jedai Stoyak lui nous tous je tu elle lui perl 10 20 20 30 8 6 9 100
  /;

my @resultat1 = doublons_hash_ano( \@listing_mots );
my @resultat2 = doublons_tranche( \@listing_mots );
my @resultat3 = doublons_grep( \@listing_mots );

print "Original : @listing_mots\n";
print "HASH anonyme : @resultat1\n";
print "Tranche : @resultat2\n";
print "GREP : @resultat3\n";

sub doublons_hash_anonyme {
  my ($ref_tabeau) = @_;

  return keys %{ { map { $_ => 1 } @{$ref_tabeau} } };
}

sub doublons_grep {
  my ($ref_tabeau) = @_;

  my %hash_sans_doublon;
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Exemple

```
    return grep { !$hash_sans_doublon{$_}++ } @{$ref_tabeau};
}

sub doublons_tranche {
    my ($ref_tabeau) = @_;

    my %hash_sans_doublon;    # Comme un hash ne peut pas avoir deux clés identiques,
                              # utilser ces clé permet d'avoir un tableau unique
    @hash_sans_doublon{ @{$ref_tabeau} } = ();
    # Pas besoin de surcharger le hash avec des valeurs inutiles
                                                # et ensuite, on renvoie le tableau des clés uniques
    return keys %hash_sans_doublon;
}

sub doublons_hash_ano {
    my ($ref_tabeau) = @_;

    return keys %{ { map { $_ => 1 } @{$ref_tabeau} } };
}
```

Résultat

```
Original : moi 2Eurocents Woufeil toi Djibril Jedai moi Jedai gldavid etc Jedai Stoyak lui nous tous
 je tu elle lui perl 10 20 20 30 8 6 9 100
HASH anonyme : Djibril toi je gldavid etc tu Woufeil 30 100 moi 6 Jedai 9 20 8 2Eurocents perl tous
 lui Stoyak elle 10 nous
Tranche : Djibril toi je gldavid etc tu Woufeil 30 100 moi 6 Jedai 9 20 8 2Eurocents perl tous lui
 Stoyak elle 10 nous
GREP : moi 2Eurocents Woufeil toi Djibril Jedai gldavid etc Stoyak lui nous tous je tu elle perl 10
 20 30 8 6 9 100
```

On peut constater que l'ordre est conservé pour la technique 3. Voilà, à vos tests. :-) !!

Comment trier un tableau ?

Auteurs : 2Eurocents ,

La fonction sort() en Perl est très puissante et permet de faire tous les tris inimaginables.
Elle trie les éléments d'une liste et retourne une liste triée des éléments fournis en paramètre.
Ces deux listes peuvent être distinctes, ou bien la liste d'origine peut être écrasée par le résultat du tri si elle est destinataire de l'affectation.
La fonction sort peut prendre un argument spécial qui est le bloc de comparaison à effectuer entre les éléments de la liste à trier. Il est ainsi possible de trier selon des critères tout à fait personnalisés.
Par défaut, le tri se fait dans l'ordre "lexicographique" (ordre alphabétique, étendu aux nombres dont les chiffres sont traités comme des caractères et non comme des nombres). Le bloc de comparaison personnalisé doit
être précisé entre accolades, avant la liste d'éléments à trier, sans être séparé de celle-ci par une virgule. Pour la comparaison, ce bloc utilise deux variables internes de PERL, $a et $b, qui ne sont définies que dans ce bloc et masquent toute variable $a ou $b propre à l'utilisateur. Ce bloc effectue donc un test quelconque, basé sur $a et $b, dont le résultat peut prendre trois valeurs :

- **positif si $a est avant $b dans l'ordre de tri souhaité**
- **nul si $a et $b sont équivalents**
- **négatif si $a est après $b dans l'ordre souhaité.**

Ces trois valeurs de résultat de test correspondent aux résultats des opérateurs de test et cmp vus précédemment.
Ainsi, pour un *tri lexicographique* du tableau @t, on peut faire :

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
@l = sort ( @t ); ou bien @l = sort ( { $a cmp $b } @t );
```

qui a l'avantage d'être totalement explicite et donc plus clair à maintenir.
Pour un *tri lexicographique inversé*, on peut aussi bien écrire :

```
@l = reverse ( sort ( { $a cmp $b } @t ) );
```

que

```
@l = sort ( { $b cmp $a } @t );
```

Mais sachez que reverse est beaucoup plus optimal que faire sort ({ $b cmp $a } @t.
Pour un *tri numérique* :

```
@l = sort ( { $a <=> $b } @t );
```

Pour un tri un peu spécial, en supposant que @t contienne des indices d'un tableau et que l'on souhaite un tri de ces indices selon les valeurs numériques du tableau :

```
@l = sort ( { $tableau[$a] <=> $tableau[$b] } @t );
```

@l contient alors les valeurs de @t triées dans un ordre tel qu'elles indicent des valeurs croissantes dans @tableau. C'est très indirect et ce n'est pas forcément très naturel au début, mais c'est extrêmement puissant. Et pour finir, bien que l'on ait pas encore abordé la définition de fonctions personnalisées, il est possible d'appeler une fonction définie par le programmeur :

```
@l = sort ( { mafonction ($a, $b) } @t );
```

Nous vous recommandons de lire les articles ci dessous

les mongueurs de Perl (groupe francophone consacré à la promotion de Perl) ont écrit un très bon article.
En voici d'autres :

L'excellent article de Uri Guttman et Larry Rosler : *A Fresh Look at Efficient Perl Sorting*,

La traduction de cet article par votre serviteur les mongeurs est disponible ici.

Comment comparer plusieurs tableaux entre eux ?

Auteurs : Djibril ,

Il arrive que l'on ait besoin de comparer 2 listes perl entre elles afin d'obtenir les données communes aux deux listes, les données présentes dans une liste mais pas dans l'autre ... Pas besoin de reinventer la roue, il existe un module sur le CPAN qui le fait très bien, c'est List::Compare. Voici un exemple de code :

```
#!/usr/bin/perl
use strict;
use warnings;
use List::Compare;

my @Llist = qw(abel abel baker camera delta edward fargo golfer);
my @Rlist = qw(baker camera delta delta edward fargo golfer hilton);

my $lc = List::Compare->new( \@Llist, \@Rlist );
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```perl
my @intersection = $lc->get_intersection;
my @union        = $lc->get_union;

print "@intersection\n@union\n";
# baker camera delta edward fargo golfer
# abel baker camera delta edward fargo golfer hilton
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Comment participer à la liste des codes unilignes ?

Auteurs : Djibril ,

Vous trouvez une erreur, vous avez des remarques à effectuer ou si vous souhaitez partager une liste de codes unilignes perl, n'hésitez pas et postez le ici :

Utilise-t-on des simples ou double quotes dans les unilignes ?

Auteurs : Djibril , Philou67430 ,

- **Sous Linux/Unix**

Sous une console Linux/Unix, vous devez utiliser des simples quotes. Exemple :

Simple quote
```
#perl -e '$pseudo = 'Djibril'; print $pseudo;'
Djibril
```

Double quote
```
C:\>perl -e "$pseudo = 'Djibril'; print $pseudo;"
syntax error at -e line 1, near "="
Execution of -e aborted due to compilation errors.
```

En fait, le programme a subi une interpolation de la variable $pseudo. Perl a donc compris = 'Djibril'; au lieu de $pseudo = 'Djibril';. S'il y avait une variable d'environnement s'appelant pseudo, Perl l'aurait interprété autrement.
Si vous tenez absolument à utiliser des guillemets, protégez votre variable avec un \.

```
# perl -e "\$pseudo = 'Djibril'; print \$pseudo;"
Djibril
```

L'écriture est un peu moins évidente et compréhensive.

- **Sous Windows**

Sous une console DOS Windows, vous devez utiliser des double quotes sous peine d'avoir un message d'erreur. Exemple :

Simple quote
```
C:\>perl -e '$pseudo = 'Djibril'; print $pseudo;'
Can't find string terminator "'" anywhere before EOF at -e line 1.
```

Double quote
```
C:\>perl -e "$pseudo = 'Djibril'; print $pseudo;"
Djibril
```

- **Sous Windows - Cygwin**

Le comportement du Shell est semblable à celui d'Unix, donc utilisez des simple quotes.

- **Astuces à connaître**

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Perl vous donne la possibilité d'utiliser 3 fonctions de remplacement des guillemets simples, doubles ou renversés : qq, q et qx.

Fonction	Remplacement	Interpolation
qq{}	"	oui
q{}	'	non
qx{}	``	oui

Exemple q{}, qq{} et qx{}

```
perl -e '$pseudo = q{Djibril}; print qq{Pseudo : $pseudo\n};'
Pseudo : Djibril

# perl -e 'print qx{ls -alh /home}'
total 1.0K
drwxr-xr-x  4 root root  104 Mar  4 08:38 .
drwxr-xr-x 21 root root  512 Mar 11 10:27 ..
drwxr-xr-x  8 Djibril Djibril 584 Mar  4 10:00 Djibril
```

Attention : Sous Unix et Cygwin, on ne peut pas utiliser des quotes simples pour délimiter des chaînes non interprétables dans un uniligne défini avec des quotes simples. Ainsi, pour :

```
perl -e '$pseudo = 'Djibril'; print $pseudo;'
```

ça fonctionne car la chaîne Djibril ne contient aucun caractère interprétable du Shell. En revanche, ça ne fonctionne plus dès lors qu'on utilise :

```
perl -e '$pseudo = 'Djibril'; print 'pseudo' ;'
```

ou encore

```
perl -e '$pseudo = 'Djibril'; print '$pseudo=', $pseudo, "\n";'
```

Si l'on utilise la quote simple pour l'uniligne, il faut impérativement utiliser q{} pour définir des chaînes littérales non interprétables :

```
perl -e '$pseudo = q{Djibril}; print q{$pseudo=}, $pseudo, "\n";'
```

Pour en savoir plus sur ces fonctions, regardez la documentation officielle :

- **man perlop sous Linux/Unix ;**
- **perldoc perlop sous Windows.**

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > Codes sources utiles > Des codes sources > Quelques unilignes perl > Traitements de fichiers

Comment supprimer les ^M dans un fichier pour le rendre compatible Unix/Mac/Linux/Windows ?

Auteurs : Djibril ,

Ce code vous permettra de convertir un fichier plat DOS en UNIX en convertissant les retours chariots vers le format unix. Cela permet de supprimer les ^M dans un fichier Mac par exemple. Cela est également pratique pour les fichiers transferés en FTP en mode binaire au lieu de ASCII. C'est léquivalent d'un dos2unix.

```
perl -pi.bak -e "s/\r\n/\n/" mon_fichier.txt
```

Ce code fait une sauvegarde préalable du fichier et supprime les ^M de mon_fichier.txt.

```
perl -pi -e "s/\r\n/\n/" mon_fichier.txt
```

Fait la même chose sans sauvegarde.

Comment supprimer les lignes doublons dans un fichier ?

Auteurs : Djibril ,

Attention : bien qu'efficace, cette méthode est consommatrice de mémoire si vous l'appliquez sur un très gros fichier car ce dernier est entièrement chargé en mémoire.

```
perl -ne "print if ! $lignes{$_}++" mon_fichier.txt > nouveau_fichier.txt
```

```
perl -ne 'print if ! $lignes{$_}++' mon_fichier.txt > nouveau_fichier.txt
```

Comment effectuer des remplacements dans un fichier ?

Auteurs : Djibril ,

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Il peut être utilise d'avoir une petite commande perl permettant de modifier un mot dans un fichier. Cet uniligne est là pour ça. Dans l'exemple ci-dessous, nous changeons le nom de domaine yahoo.fr par developpez.com dans un fichier XML.

annuaire.xml
```
<annuaire>
  <personne>
    <nom>Philou</nom>
    <prenom>Jean-Francois</prenom>
    <telephone>01234567890</telephone>
    <email>toto@yahoo.fr</email>
  </personne>
  <personne>
    <nom>Dupont</nom>
    <prenom>Paul</prenom>
    <telephone>09876543210</telephone>
    <email>titi@yahoo.fr</email>
  </personne>
</annuaire>
```

avec sauvegarde du fichier original
```
perl -pi.sauvegarde -e "s/yahoo\.fr/developpez\.com/g" annuaire.xml
```

Comment afficher des colonnes particulières depuis un fichier de type csv (tableau texte) ?

Auteurs : Philou67430 , Alek-C , Djibril ,

Windows
```
perl -ne "chomp; print join qq{ }, @{[ split /;/ ]}[8,0],$/" fichier.csv
```

Linux/Unix
```
perl -ne 'chomp; print join qq{ }, @{[ split /;/ ]}[8,0],$/' fichier.csv
```

Dans ce code, l'expression régulière du split constitue le séparateur de colonne du fichier d'entrée passé en paramètre (le retour à la ligne est considéré comme le séparateur de ligne). Les colonnes affichées dans l'exemple sont les colonnes d'indice 8 et 0, dans cet ordre. Elles sont affichées avec le paramètre du join comme séparateur.

fichier.csv
```
col0;col1;col2;col3;col4;col5;col6;col7;col8;col9;col10
test1;test2;test3;test4;test5;test6;test7;test8;test9;test10;test11
```

resultat
```
col8 col0
test9 test1
```

Il est possible de raccourcir le code via les options -a -F et -l.

Windows
```
perl -laF';' -ne "print join q( ), @F[8,0]" fichier.csv
```

Linux/Unix
```
perl -laF';' -ne 'print join qq( ), @F[8,0]' fichier.csv
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Ce code correspond à un code perl complet suivant :

```
#!/usr/bin/perl
$\ = $/;                        # option -l
while (<>) {                    # option -n
  chomp $_;                     # options -l et -n
  @F = split /;/;               # option -a et -F
  print join qq( ), @F[8,0];    # option -e
}
```

Comment inverser tous les octets d'un fichier ?

Auteurs : Djibril ,

```
perl -e "print scalar reverse <>" fichier
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Insérer une ligne dans un fichier, à une position donnée

Auteurs : Schmorgluck ,

Exemple illustratif pour l'insertion de "toto" à la ligne 4 du fichier text, avec backup du fichier d'origine en .bak.

```
perl -i.bak -lpe'print "toto" if $.==4' text
```

Insérer une ligne dans un fichier, avant chaque ligne correspondant à une regex

Auteurs : Schmorgluck ,

Exemple illustratif pour l'insertion de "toto" avant chaque ligne du fichier text correspondant à la regex /lorem ipsum/, avec backup du fichier d'origine en .bak.

```
perl -i.bak -lpe'print "toto" if /lorem ipsum/' text
```

Insérer une ligne dans un fichier, après chaque ligne correspondant à une regex

Auteurs : Schmorgluck ,

Exemple illustratif pour l'insertion de "toto" après chaque ligne du fichier text correspondant à la regex /lorem ipsum/, avec backup du fichier d'origine en .bak.

```
perl -i.bak -lne'print ; print "toto" if /lorem ipsum/' text
```

Insérer une ligne dans un fichier, après la n-ième ligne correspondant à une regex

Auteurs : Schmorgluck ,

Exemple illustratif pour l'insertion de "toto" après la quatrième ligne du fichier text correspondant à la regex /lorem ipsum/, avec backup du fichier d'origine en .bak.

```
perl -i.bak -lne'print ; print "toto" if /lorem ipsum/ and ++$occurrence==4' text
```

Insérer une ligne dans un fichier en plusieurs positions

Auteurs : Schmorgluck ,

Exemple illustratif pour l'insertion de "toto" aux première, troisième et quatrième lignes du fichier text, avec backup du fichier d'origine en .bak.

```
perl -i.bak -lpe'print "toto" if  grep {$_==$.} (1,3,4)' text
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Attention, les positions sont relatives aux anciennes positions des lignes. Puisqu'une insertion a déjà eu lieu à la première ligne, l'insertion effectuée à la troisième ligne se retrouve de fait en quatrième ligne, et ainsi de suite. Je n'ai pas trouvé de solution simple pour résoudre ce problème.

Insérer une ligne dans un fichier, après certaines occurrences de lignes correspondant à une regex

Auteurs : Schmorgluck ,

Exemple illustratif pour l'insertion de "toto" après les première, troisième et quatrième lignes du fichier text correspondant à la regex /lorem ipsum/, avec backup du fichier d'origine en .bak.

```
perl -i.bak -lne'print ; print "toto" if /lorem ipsum/ and ++$occurrence and grep {$_==
$occurrence} (1,3,4)' text
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Comment récupérer le code source d'une page web ?

Auteurs : Djibril ,

```
perl -MLWP::Simple -e "print get shift" http://www.lemonde.fr/
```

```
perl -MLWP::Simple -e "print get shift" http://www.lemonde.fr/ > lemonde.html
```

Ce code utilise le module LWP::Simple pour télécharger le code source du lien qui est donné en argument.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > Codes sources utiles > Des codes sources > Quelques unilignes perl > Divers

Comment savoir si un nombre est un nombre premier ?

Auteurs : Djibril ,

Voici comment tester si un nombre est premier (code d'Abigail).

```
perl -wle "print qq{Premier} if (1x shift) !~ /^1?$|^(11+?)\1+$/" NOMBRE
```

```
perl -wle 'print "Premier" if (1x shift) !~ /^1?$|^(11+?)\1+$/' NOMBRE
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > Codes sources utiles > Des codes sources > Terminal

Comment afficher des accents sur une console Windows (DOS) ?

Auteurs : Jedai , Djibril ,

Vous avez sans doute déjà rencontré un problème d'affichage d'accents sur une console DOS.

```
#!/usr/bin/perl
use strict;
use Carp;
use warnings;

print "J'essaie d'utiliser des accents àéé\n";
print "Essayons les entrées-sorties : \n";
```

```
J'essaie d'utiliser des accents ÓÚÚ
Essayons les entrÚes-sorties :
```

Pour palier à ce désagrément, le système de gestion des encodages de Perl est excellent, puissant et pratique, mais mal compris par pas mal de gens. Pour afficher les accents correctement sur la console DOS Windows, Il faut mettre stdout avec l'encodage correct, ou utiliser l'un des modules spécialisés. chcp permet de récupérer la codepage correcte. Premièrement, récupérons le codepage correct.

codepage DOS
```
my ($codepage) = (`chcp` =~ m/:\s+(\d+)/);
```

Encodons ensuite toutes les sorties STDOUT, STDERR et même STDIN.

```
foreach my $h ( \*STDOUT, \*STDERR, \*STDIN ) {
  binmode $h, ":encoding(cp$codepage)";
}
```

Je vous recommande de toujours créer vos scripts au mode utf8. Pour cela, vous devez configurer votre éditeur de texte. Exemple avec PSPAD, il vous suffit de choisir utf8 dans le menu "format". Votre script sera ainsi au format utf8. Ensuite, mettez en début de script :

```
use utf8;
```

Voilà, maintenant tout "print" avec accents sera correct dans votre console DOS. Voici un script qui résume les codent ci-dessus qu'on mettra dans une fonction.

```
#!/usr/bin/perl
use strict;
use Carp;
use warnings;

use utf8;

ActiverAccentDOS();

print "J'essaie d'utiliser des accents àéé\n";
print "Essayons les entrées-sorties : \n";

sub ActiverAccentDOS {
  my ($codepage) = ( `chcp` =~ m/:\s+(\d+)/ );
  foreach my $h ( \*STDOUT, \*STDERR, \*STDIN ) {
    binmode $h, ":encoding(cp$codepage)";
  }
}
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
J'essaie d'utiliser des accents àéé
Essayons les entrées-sorties :
```

C'est quand même beaucoup mieux !! Cette fonction peut être mise dans un de vos modules et appelée en début de vos scripts. Si vous ne souhaitez pas appliquer la modification d'affichage à tout STDOUT, vous pouvez toujours utiliser le module Encode.

```perl
#!/usr/bin/perl
use strict;
use Carp;
use warnings;

use utf8;
use Encode;

my ($codepage) = ( `chcp` =~ m/:\s+(\d+)/ );
print encode("cp$codepage", "J'essaie d'utiliser des accents àéé\n");
print encode("cp$codepage", "Essayons les entrées-sorties : \n");
```

Si vous ne souhaitez pas utiliser le module utf8, il est toujours possible d'utiliser les codes hexa corrects. Faut-il encore s'en souvenir ! Cette solution est moins propre et maintenable, mais peut toujours être utile.

ü	\x81	à	\x85	è	\x8A
é	\x82	ç	\x87	ï	\x8B
â	\x83	ê	\x88	î	\x8C
ä	\x84	ë	\x89	...	...

Les informations de ce tableau ont été trouvées ici, ce qui nous donne :

```perl
#!/usr/bin/perl
use strict;
use Carp;
use warnings;

print "J'essaie d'utiliser des accents \x85\x82\x82\n";
print "Essayons les entr\x82es-sorties : \n";
```

Voilà, notez quand même que ce n'est pas très lisible :-) !!

Consultez la discussion sur le forum au sujet des accents.

Comment saisir un mot de passe de façon invisible sur une console ?

Auteurs : Djibril ,

Pour saisir un mot de passe de façon invisible sur une console DOS, linux ou unix, il y a un module Perl Term::ReadPassword .

Sous windows, l'auteur du module recommande d'utiliser le module Term::ReadPassword::Win32.

windows

```perl
#!/usr/bin/perl
use strict;
use warnings;

use Term::ReadPassword::Win32;
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

windows

```
# Faire deviner le mot de passe : perl
while (1) {
  my $password = read_password('Veuillez deviner le mot de passe : ');
  redo unless defined $password;
  if ($password eq 'perl') {
    print "Bravo, c'est le bon.\n";
    last;
  }
  else {
    print "Dommage!\n";
    redo;
  }
}
```

linux/unix

```
#!/usr/bin/perl
use strict;
use warnings;
use Term::ReadPassword;

# Faire deviner le mot de passe : perl
while (1) {
  my $password = read_password('Veuillez deviner le mot de passe : ');
  redo unless defined $password;
  if ($password eq 'perl') {
    print "Bravo, c'est le bon.\n";
    last;
  }
  else {
    print "Dommage!\n";
    redo;
  }
}
```

Comment afficher une barre de progression sur une console ?

Auteurs : Philou67430 , Djibril ,

Pour afficher une barre de progression sur la console DOS, Linux ou Unix, il est possible de la créer soit même ou bien d'utiliser des modules existant. Nous allons vous donner 2 exemples.

- **Code conçu soit même**

Progressbar.pm

```
package Progressbar;

use strict;
use warnings;

# Constructor
sub new {
  my $proto = shift;
  my %p = @_;
  my $class = ref($proto) || $proto;
  my $self  =
    {
     anim              => [ ref($p{anim}) eq 'ARRAY' ? @{$p{anim}} : qw(\\ | / -) ],
     text              => $p{text}      || "",
     max               => $p{max}       || 100,
     size              => $p{size}      || 40,
     value             => $p{init}      || 0,
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Progressbar.pm

```
      display_count     => $p{verbose}   || 0,
      animation         => $p{anim}      || 0,
      fill_char         => $p{fill_char} || "#",
      nb_filled         => 0,
      last_size_printed => 0,
    };
    $self->{step} = $self->{max} / $self->{size};

    bless ($self, $class);
    return $self;
}

# Progress method
sub inc {
    my $self = shift;

    my $nb_filled = int(($self->{value}++) / $self->{step}) + 1;
    if ($self->{nb_filled} != $nb_filled || $nb_filled == $self->{size}) {
      my $pb = sprintf ("$self->{text} [%-$self->{size}s]",
                        $self->{fill_char} x $nb_filled);
      $pb .= " ".$self->{anim}->[$nb_filled % @{$self->{anim}}] if $self->{animation};
      $pb .= sprintf " %3d/$self->{max}", $self->{value} if $self->{display_count};
      $pb .= "\r";

      print STDERR $pb;
      $self->{last_size_printed} = length $pb;
    }
    $self->{nb_filled} = $nb_filled;
}

# Erase method
sub erase {
    my $self = shift;

    my ($string) = @_;

    print STDERR (" " x $self->{last_size_printed})."\r";
    print STDERR $string if defined $string;
}

# Reset/end method
sub end {
    my $self = shift;

    $self->{value}     = 0;
    $self->{nb_filled} = 0;
    print STDERR "\n";
}

1;

__END__

=head1 NAME

=head1 SYNOPSIS

use Progressbar;

my $pb = Progressbar->new(text => "Computing");

foreach (0 .. 99) {
  # ...
  $pb->inc;
}
$pb->erase("Process done");
$pb->end; # $pb can be re-used immediatly
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Progressbar.pm

```
my $pb2 = Progressbar->new(text => "Working", anim => 1);
my $pb3 = Progressbar->new(text => "Working",
                           max  => 1000,
                           size => 58,
                           init => 500,
                           verbose => 1,
                           anim => [qw(0 1 2 3 4 5 6 7 8 9)],
                           fill_char => "=");

foreach (0 .. 99) {
  # ...
  $pb2->inc;
}
$pb2->end;

foreach (500 .. 999) {
  # ...
  $pb3->inc;
}
$pb3->end;

=head1 DESCRIPTION

Prints a progress bar on STDERR using ASCII characters.

=head1 BUGS

No known bug.
```

Pour l'utiliser, créez un script test.pl par exemple

test.pl

```
#!/usr/bin/perl

use strict;
use warnings;

use lib ".";
use Time::HiRes qw(sleep);

use Progressbar;

my $pb = Progressbar->new(text => "Computing");

foreach (0 .. 99) {
  sleep(rand(0.1));
  $pb->inc;
}
$pb->erase("Process done");
$pb->end; # $pb can be re-used immediatly

my $pb2 = Progressbar->new(text => "Working", anim => 1, verbose => 1);
my $pb3 = Progressbar->new(text => "Working",
                           max  => 1000,
                           size => 58,
                           init => 500,
                           verbose => 1,
                           anim => [qw(0 1 2 3 4 5 6 7 8 9)],
                           fill_char => "=");

foreach (0 .. 99) {
  sleep(rand(0.1));
  $pb2->inc;
}
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

test.pl
```
$pb2->end;

foreach (500 .. 999) {
  sleep(rand(0.02));
  $pb3->inc;
}
$pb3->end;
```

- **Utilisation d'un module existant sur le CPAN Term::ProgressBar**

test2.pl - module CPAN Term::ProgressBar
```
#!/usr/bin/perl
use warnings;
use strict;
use Term::ProgressBar;
use Time::HiRes qw(sleep);

my $MaxValue = 100;
my $barre_progression1 = Term::ProgressBar->new(
    { name  => "[Computing] ",
      count => $MaxValue,
      ETA   => "linear",
    }
);
foreach (0 .. $MaxValue) {
  sleep(rand(0.1));
  $barre_progression1->update($_);
}
$barre_progression1->message("Process done");

my $barre_progression2 = Term::ProgressBar->new(
    { name  => "[Working] ",
      count => $MaxValue,
    }
);
foreach (0 .. $MaxValue) {
  sleep(rand(0.1));
  $barre_progression2->update($_);
}
$MaxValue = 1000;
$barre_progression2->target($MaxValue);
$barre_progression2->message('max: 1000');
foreach (0 .. $MaxValue) {
  sleep(rand(0.01));
  $barre_progression2->update($_);
}

print "\nFini!";
```

L'utilisation d'une barre de progression peut être intéressant lorsque que vous lisez un très gros fichier. Ca vous permet de voir l'avancement de la lecture du fichier. Pour faire cela, il suffit de récupérer la taille du fichier via stat afin de créer sa barre de progression, puis au fur et à mesure de la lecture du fichier, de mettre à jour la barre avec la taille courante du fichier en cours de lecture.
Exemple :

Lecture d'un fichier avec barre de progression
```
#!/usr/bin/perl
use warnings;
use strict;

use Term::ProgressBar;
my $file     = 'mon_fichier.txt';
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Lecture d'un fichier avec barre de progression

```
my $size_max = ( stat $file )[7];
my ( $compteur_modif, $progression_octet, $next_update ) = ( 0, 0, 0 );
my $barre_progression = Term::ProgressBar->new(
    {   name  => "[Lecture $file] ",
        count => $size_max,
        ETA   => "linear",
    }
);

open( my $fh, '<', $file ) or die "Impossible d'ouvrir le fichier $file\n";
while ( my $ligne = <$fh> ) {

    # ... <=== on travaille sur la ligne
    #bar
    $progression_octet += length($ligne);
    $next_update = $barre_progression->update($progression_octet)
        if ( $progression_octet > $next_update
        && $progression_octet < $size_max );
}
close($fh);

$barre_progression->update($size_max) if $size_max >= $next_update;
```

Comment créer une question à choix multiple dans un terminal ?

Auteurs : Djibril ,

Si vous avez besoin que l'utilisateur interagisse avec votre script, vous avez le choix de le faire via une interface graphique (Perl/Tk, gtk2, etc), ou via la console (DOS, Linux, Unix, ...).
Pour interagir via une console, on utilise généralement <STDIN>. C'est simple, mais lorsque l'on a plusieurs choix, on est obligé de faire plusieurs boucles afin de tester la validité du choix. De plus, il ne faut pas oublier de faire un chomp pour éliminer les retour chariots. Il existe une multitude de modules sur le CPAN nous facilitant grandement la vie.
C'est le cas du module Term::UI.

Voici un exemple de code nécessitant peu de lignes

```
#!/usr/bin/perl
use warnings;
use strict;

use Term::UI;
use Term::ReadLine;

my $term    = Term::ReadLine->new('prompt');
my $reponse = $term->get_reply(
  prompt  => 'Choisir un nombre : ',
  choices => [ 'Choix 1', 'Choix 2', 'Choix 3', 'Aucun choix'],
  default => 'Aucun choix',
);

print "Vous avez choisi : $reponse\n";

my $bool = $term->ask_yn(
                        prompt => 'Aimez vous les cookies ?',
                        default => 'n',
                );
if ( $bool ) {
  print "J'aime les cookies\n";
}
else {
  print "Je n'aime pas les cookies\n";
}
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

On obtient le résultat suivant :

```
1> Choix 1
  2> Choix 2
  3> Choix 3
  4> Aucun choix


Choisir un nombre :  [4]: 7

Invalid selection, please try again: [4] 2
Vous avez choisi : Choix 2

Aimez vous les cookies ? [y/N]: y
J'aime les cookies
```

Via la méthode get_reply, vous pouvez simuler un QCM facilement et proprement. Et si vous souhaitez poser une question attendant une réponse yes or no, c'est encore plus simple avec la méthode ask_yn.
Pour en savoir plus, la documentation CPAN est à votre disposition.

Comment récupérer proprement les arguments de la ligne de commande ?

Auteurs : Djibril ,

Lorsque vous souhaitez appeler vos scripts perl via une console DOS ou Linux (ou Unix), vous avez souvent besoin de passer des arguments. La manière classique pour les récupérer est d'utiliser la variable @ARGV. Sachez qu'il existe un module perl nous permettant de gérer ces arguments proprement et facilement. Ce module est de plus déjà installé dans le core de perl, c'est le module Getopt::Long. Voici un exemple de code nous permettant de taper en ligne de commande ceci :

```
perl script.pl -nom Djibril -fichier "C:\repertoire\fichier.txt"
```

```
#!/usr/bin/perl
use strict;
use warnings;
use Getopt::Long;

my ( $nom, $fichier ) = ();
GetOptions( 'nom=s' => \$nom, 'fichier=s' => \$fichier,);

if ( defined $nom ) {
  print "nom : $nom\n";
}
if ( defined $fichier ) {
  print "fichier : $fichier\n";
}
```

résultat

```
nom : Djibril
fichier : C:\repertoire\fichier.txt
```

Ce module est très complet, n'hésitez pas à lire la documentation officielle. Vous pouvez créer des alias, récupérer plusieurs arguments pour une même option, etc. Voici un dernier exemple pour la route !!

```
perl script.pl -name Djibril -fichier "C:\repertoire\fichier.txt" -fichier "D:\repertoire
\fichier2.txt" -numero 2 -v
```

```
#!/usr/bin/perl
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```perl
use strict;
use warnings;
use Getopt::Long;

my ( $nom, @fichiers, $numero, $verbeux) = ();
GetOptions(
  'nom|name=s' => \$nom,      # name ou nom
  'fichier=s'  => \@fichiers, # On peut avoir plusieurs fichiers
  'numero=i'   => \$numero,   # i pour integer => on récupère un entier
  'verbeux|v'  => \$verbeux,  # flag de type -v ou -verbeux ou --v ou --verbeux
);

if ( defined $nom ) {
  print "nom : $nom\n";
}
if ( defined $numero ) {
  print "numero : $numero\n";
}
if ( @fichiers ) {
  print "fichiers : @fichiers\n";
}
if ( $verbeux ) {
  print "mode verbeux\n";
}
```

résultat

```
nom : Djibril
numero : 2
fichiers : C:\repertoire\fichier.txt D:\repertoire\fichier2.txt
mode verbeux
```

NB: Notez l'importance de protéger le chemin de vos fichiers par des guillemets, surtout s'il y a des espaces. Voilà, amusez vous bien !!

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > Codes sources utiles > Des codes sources > Web

Comment récupérer une page Web ?

Auteurs : Dimitry.e ,

Voici un script permettant de récupérer le contenu d'une page Web et l'afficher sur la sortie standard STDOUT. Il y a la possibilité de spécifier un proxy HTTP. Ainsi qu'un login pour les connexions HTTP (autorisation de base).

```
#!/usr/bin/perl
use warnings;
use strict;
use Getopt::Long;
use LWP::Simple;
use LWP::UserAgent;
use MIME::Base64;

=for COMM

Script permettant de recuperer le contenu du page web en HTTP
avec utilisation d'un proxy et d'un login si besoin.

Arguments obligatoires :
  --url=URL
    Precise la page a recuperer

Arguments Optionnels :
  --login=USER:PASSWD
    Specifie le login a utiliser
  --proxy=PROXY
    Indiquer un proxy HTTP

=cut

my ( $url, $login, $proxy );

# Recuperation et stockage des options dans leurs variables respectives
GetOptions(
  'url=s'   => \$url,
  'login=s' => \$login,
  'proxy=s' => \$proxy,
);

# Le script a besoin d'au moins une URL pour fonctionner
if ( !defined $url ) { die( 'Usage : ' . $0 . " --url=URL\n" ); }

# Le login doit etre forme du nom et du mot de passe separes par deux points
if ( ( defined $login ) and ( $login !~ /.+:.+/ ) ) {
  die( 'Usage : ' . $0 . " --url=URL --login=USER:PASSWD\n" );
}

# Creation du User Agent. Il se charge de traiter la requete HTTP, comme un navigateur le ferait.
my $ua = LWP::UserAgent->new;

# Le timeout permet de savoir a partir de quand on considere qu'un requete n'aboutira pas
$ua->timeout(20);

# On crée la requete HTTP correspondant a l'url
my $req = HTTP::Request->new( GET => $url );

if ( defined $proxy ) {

  # Indique a l'user agent qu'il va devoir utiliser un proxy
  $ua->env_proxy;

  # Indique a l'user agent quel proxy utiliser
  $ua->proxy( ['http'], $proxy );
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

```
}

if ( defined $login ) {

  # Encodage en base 64 comme le demande HTTP
  my $token = encode_base64($login);

  # Ajout de l'option 'Authorization Basic' et du login dans la requete. (cf RFC de HTTP)
  $req->header( Authorization => 'Basic ' . $token );
}

# Envoi de la requete et reception de la reponse dans $content
my $content = $ua->request($req);

# Si la requete a abouti, afficher le contenu de la page web
if ( $content->is_success ) {
  print $content->decoded_content;
}
else {
  die $content->status_line;    # Afficher la raison de l'erreur
}
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Comment faire un sleep de moins d'une seconde ?

Auteurs : Djibril ,

Pour faire un sleep de moins d'une seconde, il est recommandé d'utiliser le module Time::HiRes qui est disponible dans le core.

```
#!/usr/bin/perl
use strict;
use warnings;

use Time::HiRes qw ( sleep );
print "Bonjour, je dors\n";
sleep 0.5; # pause d'une demie seconde
print "Au revoir\n";
```

Mais vous pouvez être encore beaucoup plus précis. A vous de jouer avec ce module très puissant.

Comment désinstaller un module ?

Auteurs : Djibril ,

Sous Linux ou Mac OS, il n'existe aucun utilitaire permettant de désinstaller un module Perl. Donc, voici un script qui peut vous permet de le faire proprement.

```
#!/usr/bin/perl
use strict;
use warnings;

use ExtUtils::Packlist;
use ExtUtils::Installed;

$ARGV[0] or die "Usage: $0 Module::Name\n";

my $mod = $ARGV[0];

my $inst = ExtUtils::Installed->new();

foreach my $item (sort($inst->files($mod))) {
print "suppression de $item\n";
unlink $item;
}

my $packfile = $inst->packlist($mod)->packlist_file();
print "suppression de $packfile\n";
unlink $packfile;
```

Sous Windows, vous pouvez désinstaller un module en utilisant ppm si vous utilisez activePerl. Néanmoins, ce script peut être aussi utilisable sous Windows.

```
ppm uninstall MonModule
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

NB: **Si vous avez installé un module avec fink (sous Mac OS) ou apt-get, aptitude (Debian), yum (Red Hat), etc, utilisez ces mêmes utilitaires pour désinstaller vos modules. Il se peut que le script ci-dessus ne trouve pas le script car le répertoire d'installation lui est inconnu.**

Comment valider un numéro ISBN ?

Auteurs : Djibril ,

Le module Business::ISBN permet de tester la validité d'un numéro ISBN. Il permet également de faire des conversions isbn10 en isbn13 et vice versa. Il est également possible d'afficher le numéro isbn sous différentes formes et de trouver le code publié du numéro ISBN, le pays etc... Vous pouvez même créer un code barre contenant le numéro ISBN. Pour en savoir plus, il vous suffit de lire la documentation du module.
Voici un petit exemple (vous devez installer le module Business::ISBN et GD::Barcode)

ISBN camel book

```
#!/usr/bin/perl
use strict;
use warnings;
use Business::ISBN;

my $isbn_camelbook = '2841771407';
my $isbn           = Business::ISBN->new($isbn_camelbook);

# Test validite
if ( $isbn->is_valid ) {
  print "ISBN : $isbn_camelbook valide\n";
}
else {
  die("ISBN : $isbn_camelbook invalide\n");
}

# convert
print "ISBN10 : ", $isbn->as_isbn10->as_string, "\n";
print "ISBN13 : ", $isbn->as_isbn13->as_string, "\n";

print "ISBN STRING : ", $isbn->as_string, "\n";

#print the ISBN with hyphens at specified positions.
#this not does affect the default positions
print "Autre ecriture : ", $isbn->as_string( [] ), "\n";

#print the group code or publisher code
print "Groupe code : ", $isbn->group_code,     "\n";
print "Groupe : ",      $isbn->group,          "\n";
print "Code publie : ", $isbn->publisher_code, "\n";

# création d'un code barre de type EAN13 au format png
open( PNG, '>', 'codebarre_isbn.png' );
binmode PNG;
print PNG $isbn->png_barcode;
close(PNG);
```

Résultat

```
ISBN : 2841771407 valide
ISBN10 : 2-84177-140-7
ISBN13 : 978-2-84177-140-0
ISBN STRING : 2-84177-140-7
Autre ecriture : 2841771407
Groupe code : 2
Groupe : French speaking area
Code publie : 84177
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Voici l'image du code barre générée

ISBN camel book

Comment vérifier qu'un numéro SIRET est bien formaté ?

Auteurs : Djibril ,

Le numéro SIRET est un identifiant d'établissement. Cet identifiant numérique de 14 chiffres est articulé en deux parties : la première est le numéro SIREN de l'entreprise (ou unité légale ou personne juridique) à laquelle appartient l'unité SIRET ; la seconde, habituellement appelée NIC (Numéro Interne de Classement), se compose d'un numéro d'ordre à quatre chiffres attribué à l'établissement et d'un chiffre de contrôle, qui permet de vérifier la validité de l'ensemble du numéro SIRET.

Il existe un module Business::FR::SIRET permettant de vérifier que le numéro siret est proprement formaté, il ne vérifie absolument pas que la compagnie est existante. Pour cette dernière vérification, il existe des sites web vous permettant de faire la faire.
voici un code d'exemple :

```perl
#!/usr/bin/perl
use strict;
use warnings;
use Business::FR::SIRET;
my $numero_siret = '54209790203949';
my $c            = Business::FR::SIRET->new($numero_siret);
print "SIRET ", $c->siret() . " est valide\n" if $c->is_valid();
```

Résultat

```
SIRET 54209790203949 est valide
```

Comment tester la validité d'une adresse électronique ?

Auteurs : Djibril ,

Pour tester la validité d'une adresse électronique, inutile de s'acharner à trouver la bonne expression régulière. Il existe des modules perl le faisant pour nous proprement.

Voici un exemple de codes utilisant des modules différents Email::Valid et Mail::CheckUser

```perl
#!/usr/bin/perl
use strict;
use warnings;

use Email::Valid;

my $email = 'toto@my_google.fr';
if ( Email::Valid->address($email) ) {
  print "Email::Valid => $email valide\n";
}
else {
  print "Email::Valid => $email no, valide\n";
}
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Résultat

```
Email::Valid => toto@my_google.fr no, valide
```

```perl
#!/usr/bin/perl
use strict;
use warnings;
use Mail::CheckUser qw(check_email);

$Mail::CheckUser::Timeout = 5;
if ( check_email($email) ) {
  print "Mail::CheckUser => E-mail address <$email> is OK\n";
}
else {
  print "Mail::CheckUser => E-mail address <$email> isn't valid:\n";
}
```

Résultat

```
Mail::CheckUser => E-mail address <toto@my_google.fr> isn't valid:
```

Voilà !!

Comment raboter un texte trop long et ajouter 3 petits points ?

Auteurs : vil-farfadet ,

```perl
#!/usr/bin/perl
use strict;
use warnings;

=for rabot
La fonction rabot() rabote une chaîne de caractère à une longueur donnée si la longueur initiale est supérieure
Entrées obligatoires : Chaîne de caractères et Longueur
Entrée optionnelle   : Booléen (ne coupe pas les mots si vrai)
Sortie               : Chaîne de caractères résultante
=cut

print rabot( 'Ceci est un premier exemple de phrase.', 20 ) . "\n";
print rabot( 'Ceci est un second exemple de phrase.', 20, 1 ) . "\n";

sub rabot {
  my ( $texte, $taille, $coupe ) = @_;
  unless ($coupe) { $coupe = 0; }

  if ( length($texte) > $taille ) {

    # Variante HTML : $suspension = '&hellep;';
    my $suspension = '...';
    my $ajustement = length($suspension);
    $texte = substr( $texte, 0, $taille - $ajustement );
    $texte =~ s/([^ ,;:\.!\?\(\)])[ ,;:\.!\?\(\)]+[^ ]*$/$1/ if ($coupe);
    if ($ajustement) { $texte = $texte . $suspension; }
  }

  return $texte;
}
```

```
Ceci est un premi...
Ceci est un...
```


Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > Codes sources utiles > Téléchargements

Comment télécharger ou uploader un code dans la rubrique Perl

Auteurs : Djibril ,

Une Source application de téléchargements est mise à votre disposition.
Elle remplace avantageusement les pages sources que vous connaissez bien car vous pouvez, à présent, noter les fichiers téléchargés (en leur donnant un vote positif ou négatif) et réagir directement sur ce forum.
Outre les fichiers sources, vous pouvez également télécharger plusieurs outils de développement gratuits.
L'application vous offre également la possibilité de proposer vos propres sources.
Une fois identifié(e), uploadez votre archive ou votre fichier source : une annonce sera automatiquement postée sur ce forum et toute la communauté pourra immédiatement réagir.

L'application vous permet :
- **de faire un lien vers une page de téléchargement (par exemple pour un logiciel) ;**
- **d'uploader un fichier à télécharger ;**
- **de proposer un code source dans votre langage préféré.**

Dans tous les cas, il vous est possible de lier votre proposition à une discussion du forum afin de permettre aux visiteurs de commenter / donner leur avis.
Il n'est pas nécessaire de créer la discussion avant de proposer une source : si le champ "Utiliser un topic forum existant" est vide, une discussion sera créée automatiquement dans ce forum.

Vous souhaitez poster un code source :
- **cliquez sur Ajouter ;**
- **remplissez le formulaire. Vous devez sélectionner l'option "Proposer directement un code source" et de copier coller votre code dans la zone de saisie.**

ATTENTION : si le code que vous proposez n'est pas de vous, il est impératif que vous vous assuriez au préalable de l'accord de l'auteur avant de proposer votre source !
Vous avez la possibilité de préciser différentes rubriques liées à votre source et d'y associer une image.
Vous pouvez toujours éditer toutes ces données par la suite, si vous êtes identifié(e).
Lorsque vous validez le formulaire pour la première fois, un topic d'annonce est créé automatiquement sur ce forum.
Toute source qui nous semblera non conforme (plagiat, piratage, pub, etc.) sera immédiatement supprimée !
Merci de votre participation à cette nouvelle fonctionnalité !

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Sommaire > Divers

Débutants ou expérimentés : Comment écrire du bon code en Perl ?

Auteurs : Jedai ,

Version courte :
Mettez

```
#!/usr/bin/env perl
use strict;
use warnings;
```

au début de tous vos scripts.

Indentez proprement, ou demandez à perltidy de le faire pour vous.

```
perltidy -b mon_script.pl
```

- **éventuellement demandez à perlcritic de commenter votre code, vous n'êtes pas obligé de corriger tout ce qu'il vous dit, mais ce sera déjà un bon point de départ ;**
- **utilisez les modules du CPAN ! Ce conseil vaut doublement pour les modules du CORE (qui viennent en standard avec Perl) excepté Switch sous Perl 5.8 ;**
- **n'utilisez pas les modules pour ce que vous pouvez faire en deux lignes ;**
- **utilisez les hashs, pas les tableaux, lorsque les données ont un index textuel évident pour votre utilisation desdites données dans votre script.**

Bien sûr les conseils habituels à la programmation en n'importe quel langage s'appliquent :

- **donnez des noms significatifs à vos variables ! Ne basez pas vos noms sur le type du contenu, mais sur son utilisation, sa provenance et sa destination ;**
- **adoptez des conventions pour nommer vos variables et fonctions : en Perl la convention la plus répandue est d'utiliser des soulignés "_" entre les mots, et de commencer les noms de fonctions par un verbe. L'important est que vous (et votre équipe) adoptiez UNE convention (quelle qu'elle soit !) et que vous vous y teniez ;**
- **découpez votre code en fonctions : DRY, Don't Repeat Yourself (Ne vous répétez pas !) ;**
- **découpez votre application en modules ;**
- **n'utilisez pas goto (sauf la version "goto function" spécifique au Perl, si vous savez ce que vous faites) ;**
- **n'utilisez les variables globales que contraints et forcés ou pour des variables de configuration. Déclarez vos variables avec my().**

Version longue (aka "Les explications") :
La devise de Perl est TIMTOWDI, c'est-à-dire "There Is More Than One Way To Do It", en français : "Il y a plus d'une façon de le faire !".
Et effectivement Perl est un langage très flexible et puissant dans lequel vous pouvez adopter une multitude de méthodes différentes pour parvenir au même résultat.
Malheureusement, cela ne signifie pas que toutes les méthodes sont également viables, également efficaces, ou également faciles à entretenir ou lisibles...
Dans ce topic, je vais donc indiquer quelques points qui me semblent importants pour obtenir un résultat potable. Ce sujet est destiné aussi bien aux novices qu'aux expérimentés (je ne dirais pas aux experts, il n'y a pas beaucoup d'"experts" en Perl, et à mon avis ceux qui le sont doivent déjà savoir tout ce que je vais dire).
Le pragma "strict"
Qu'est-ce qu'un pragma ? C'est une instruction au compilateur/interpréteur qui modifie le comportement du langage sur une portée limitée. En Perl les pragmas sont chargés avec "use" ou déchargés avec "no", leur nom est tout en minuscule, contrairement aux noms de modules qui par convention doivent commencer par une majuscule.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Le pragma "strict" interdit l'usage d'un certain nombre de constructions normalement valables en Perl mais qui sont responsables de la plupart des erreurs basiques que peuvent rencontrer un novice, voire parfois un expert.
Pourquoi ces constructions ne sont-elles pas simplement supprimées du langage si elles sont si dangereuses me demanderez-vous. Pour trois raisons :

1. **le poids de l'histoire : Perl est un "vieux" langage, qui a commencé ses jours comme une simple glue Unix, alors que les shells étaient encore plus rudimentaires qu'aujourd'hui, il a retenu certains des défauts des shells de l'époque, défauts peu importants pour un petit script de quelques lignes, beaucoup plus pour un programme constitué d'une vingtaine de modules... Mais l'une des fiertés de Perl est sa compatibilité rétroactive (backward compatibility), autrement dit sa capacité à faire tourner des scripts vieux de 10 ans sans modification. Ce trait est assez rare pour le type de langage que Perl représente et il y réussit effectivement de façon impressionnante ;**
2. **l'utilité de ces constructions : en effet certaines de ces constructions peuvent se révéler utiles dans des cas très particuliers et pour faire des choses relevant de la magie noire. Mais ceci ne vous concerne que si vous avez l'habitude de mettre des modules complexes sur le CPAN, qui touchent au comportement même de Perl ou ce genre de chose ;**
3. **dans un très petit script (ne dépassant pas dix lignes à mon avis, mais d'autres seront plus larges dans leur appréciation) ou un uniligne utiliser ces constructions interdites ne porte pas vraiment à conséquence et se les interdire alourdirait inutilement le programme.**

Mais quels sont donc les effets concrets de strict ? Sans rentrer dans le détail, les principaux effets sont l'obligation de déclarer toutes ses nouvelles variables (avec my() pour les variables lexicales, our() pour les variables globales de paquetage), l'interdiction des mots "nus" (barewords, autrement dit, sans 'strict' Perl interprète tout mot isolé qu'il ne reconnait pas (pas un nom de fonction ou de HANDLE de fichier) comme une chaîne de caractères), et l'interdiction des références symboliques (l'utilisation d'une chaîne de caractères variable comme nom de variable).
Le pragma "warnings"
A partir de Perl 5.6 vous pouvez utiliser ce pragma en remplacement de l'option en ligne de commande -w (qu'on pouvait également indiquer sur la ligne de shebang d'un script). Il active les avertissements de Perl, qui vous indiquent les constructions douteuses ou ressemblant à des erreurs classiques dans votre code, à l'exécution comme à la compilation. Il ne s'agira pas toujours d'erreurs réelles, mais souvent un meilleur style de programmation peut faire disparaître ces avertissements, et dans les rares cas où vous vous estimez justifié dans votre utilisation, un autre avantage du pragma warnings apparaîtra : vous pouvez le désactivez sélectivement sur une portion de code donnée et pour certains types d'avertissements précisément, lisez "perldoc perllexwarn" pour une explication en profondeur des avantages du pragma sur l'option -w.
Perl::Tidy
Le module Perl::Tidy et l'utilitaire perltidy qui vient avec ce module permettent de reformater aisément son code Perl, par exemple pour le mettre en conformité avec une norme d'entreprise ou un format que vous préférez à celui employé dans le code d'origine. Le formatage est complètement configurable et vous pouvez ainsi façonner le résultat exactement à votre goût. Indispensable !!
Perl::Critic
Perl::Critic et l'utilitaire perlcritic associé critique votre code selon un certain nombre de règles de bonne pratique décrites sous forme de module, dont beaucoup sont distribuées avec Perl::Critic (mais pas toutes), vous pouvez configurer exactement lesquelles prendre en compte (en plus du système de "niveau de sévérité" intégré à Perl::Critic) ou même en écrire de nouvelles.

En complément de cela si vous avez le temps (ou les sous), lisez "Perl Best Practices" ou sa traduction française "De l'art de programmer en Perl", excellent bouquin, qui se lit facilement, (mais un peu cher) et qui a servi de base à perlcritic.
Les modules et le CPAN
Ne réinventez pas la roue : surtout si vous êtes débutant (et que vous n'êtes pas en train de coder dans un but pédagogique), le code que vous produirez sera sans doute moins bon, moins robuste, moins efficace que le module du CPAN correspondant, et en plus il sera plus long à réécrire. Pensez toujours au CPAN, faites vos recherches dessus avec le site officiel ou le CPAN pour Win32 (qui vous indiquera où trouver vos paquetages ppm).

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.

Ce qui ne veut pas dire que votre programme doit ressembler à une collection de modules du CPAN ! N'utilisez pas un module pour une seule fonction facilement refaite en quelques lignes par vous-même (n'oubliez pas néanmoins que parfois la fonction du module a été écrite pour effectuer cette tâche bien plus rapidement et de façon bien plus robuste que votre pauvre petit remplacement). Ceci ne vaut pas pour les modules du CORE, ils sont disponibles partout (ou presque, voir la question FAQ Comment savoir si un module est disponible dans le Core de Perl ?) et sont la plupart du temps très bien codés donc n'hésitez surtout pas à les utiliser (sauf Switch, oubliez cette abomination !! Si vous voulez un switch..case en Perl, utilisez Perl5.10 qui offre un très bon switch intégré. Les hashs sont souvent également une excellente alternative à la plupart des usages de switch..case).

Pour en savoir plus sur l'utilisation de Switch, lisez cet : FAQ item.

Il y a également le cas où vous n'êtes pas tout à fait satisfait des modules disponibles sur le CPAN, n'hésitez pas alors à vous lancer dans la création d'un nouveau module (si vous êtes un programmeur suffisamment expérimenté), c'est ainsi qu'on fait progresser les choses ! Vous pouvez également proposer un patch à l'auteur d'un des modules du CPAN : s'il améliore les choses, ne brise pas la compatibilité arrière et relève vraiment du module, il sera très certainement accepté.

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright ® 2012 Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts. Cette page est déposée à la SACD.