

Le langage Perl



Sommaire

Présentation	5
<i>Historique.....</i>	5
<i>Principales caractéristiques du langage</i>	5
<i>Quelques logiciels écrits en Perl.....</i>	5
<i>Les outils de développement</i>	5
<i>Les éditeurs</i>	5
<i>Autres outils</i>	5
<i>Documentation.....</i>	5
La gestion des packages.....	7
<i>Perl Package Manager.....</i>	7
<i>La commande Cpan</i>	8
<i>Recherche de package.....</i>	8
Eléments du langage	9
<i>Hello The World !.....</i>	9
<i>Forme simple.....</i>	9
<i>Forme élaborée</i>	9
<i>Commentaires.....</i>	9
<i>Exécution d'un script Perl</i>	9
<i>Interactivité : <STDIN></i>	9
<i>Les variables</i>	9
<i>Les nombres</i>	9
<i>Les chaînes de caractère</i>	10
<i>Les tableaux</i>	11
<i>Opérateurs.....</i>	12
<i>Arithmétiques</i>	12
<i>Concaténation</i>	12
<i>Utilisation des opérateurs arithmétiques</i>	12
<i>Comparaison</i>	13
<i>Booléens</i>	13
<i>Bit à bit</i>	13
<i>Fonctions</i>	13
<i>Arithmétiques</i>	13
<i>Chaînes</i>	13
<i>Tableau</i>	13
<i>Les tables de hachage ou tableaux associatifs</i>	14
<i>Affectations successives.....</i>	15
<i>Structures de contrôle</i>	15
<i>Conditionnelles</i>	15
<i>Répétitives</i>	15
<i>Gestion des erreurs</i>	16
Les fonctions	17
<i>Fonction simple</i>	17
<i>Récurtivité.....</i>	17
<i>Valeurs de retours</i>	17
Les fichiers.....	18
<i>Opérateurs sur fichiers.....</i>	18
<i>Test d'existence d'un fichier</i>	18
<i>Liste des fichiers du répertoire courant</i>	18
<i>Ouverture de fichier</i>	18
<i>Lecture du contenu de fichiers texte.....</i>	19
<i>Méthode de lecture universelle</i>	19
<i>Méthode de lecture avec getc</i>	19
<i>Fonctions</i>	19
Les expression régulières.....	21
<i>Caractères spéciaux</i>	21
<i>Cardinalités</i>	21

Opérateurs.....	21
Recherche et remplacement	21
Variables.....	21
Opérateur tr ou y	22
Les modules.....	23
Les modules	23
Chemins vers les modules.....	23
Utilisation des modules courants	23
Mon premier module	23
Création du module MesOutils.pm	23
Utilisation du module.....	24
Utilisation des noms composés.....	24
Structure de répertoire.....	24
Utilisation.....	24
Fonctions locales aux modules	24
Instructions par défaut.....	25
Export des symboles	25
Exportation.....	25
Utilisation des symboles	25
Méthode AUTOLOAD	25
Documentation.....	26
Appels système.....	27
Connâître la version de l'OS.....	27
Récupérer la sortie d'une commande système.....	27
Utilisation de la commande open	27
Exécution d'un bloc d'instructions.....	27
Exécution et sortie de script	27
Commandes.....	27
Les rapports.....	28
Champs utilisés par write.....	28
Champs utilisés par printf.....	28
Utilisation de write	28
Variables utilisables avec write	29
Utilisation de printf	29
Introduction au module DBI de Perl	30
La table d'exemple	30
Exécuter une requête SELECT.....	30
Saisie de données	31
Programmation objet	32
Création d'une classe	32
Utilisation de la classe.....	33
Le Perl et le Web	34
Les manières d'écrire une page Html	34
Ecrire une page Web.....	34
Script CGI	34
Utilisation du module CGI.....	34
Exécution par SSI.....	34
Gestion des formulaires	34
Logon.html.....	34
logon.pl	35
traite.pl	35
Redirection	36
Cookie	36
Création d'un cookie.....	36
Lire le cookie	36
Variables d'environnement	36
Références	37

Biographie 37
Sites 37

Présentation

Historique

Né en 1987, Perl est l'émanation d'un linguiste, Larry Wall. Son objectif était de fournir un langage combinant le meilleur du C, de Awk, de Sed et du Shell.

Larry Wall a donné deux rétronymes pour qualifier le mieux son langage :

- Practical Extraction and Report Language ou langage pratique d'extraction et de génération de rapports
- Pathetically Eclectic Rubbish Lister ou collectionneur de déchets pathétiquement éclectique

Principales caractéristiques du langage

La syntaxe du langage est très proche de celle du Php.

- Il s'agit d'un langage interprété.
- Il est très utilisé dans le maniement des chaînes de caractères.
- Les variables ne sont pas typées.
- Il est objet et procédural.
- Il manie les fichiers binaires de façon transparente.

Quelques logiciels écrits en Perl

Citons, parmi les projets les plus connus aujourd'hui sous Linux :

- Sympa, gestionnaire de listes de diffusion
- SpamAssassin, filtre courriel anti-spam
- Bugzilla, gestionnaire de bugs
- Webmin, logiciel de configuration d'une machine Linux via une interface web
- Drakconf, le configurateur Mandriva
- Urpmi, le gestionnaire de packages Mandriva
- AWStats, analyseur de logs des serveurs Web

Les outils de développement

Les éditeurs

Produit	Windows	Linux
EngInSite Perl Editor Professional	Payant	
ActiveState Komodo	Payant	
Open Perl Ide	Gratuit	
OptiPerl	Payant	
Perl Builder	Payant	
PerlEdit	Gratuit	Gratuit
Perl Oasis	Gratuit	
Scite	Gratuit	Gratuit
PerlComposer		Gratuit
Visual Perl	Gratuit	
Perl Express	Gratuit	
EngInSite Perl Editor Lite	Gratuit	
Perl EDI Suite		Gratuit
DzSoft Perl Editor	Gratuit	
PerlWiz	Gratuit	

Autres outils

Prima	boîte à outils intégrant des composants Perl
wxGlade	Concepteur d'interfaces
Perl2exe	Outil de conversion de scripts Perl en exécutables
wxDesigner	Concepteur d'interfaces

Documentation

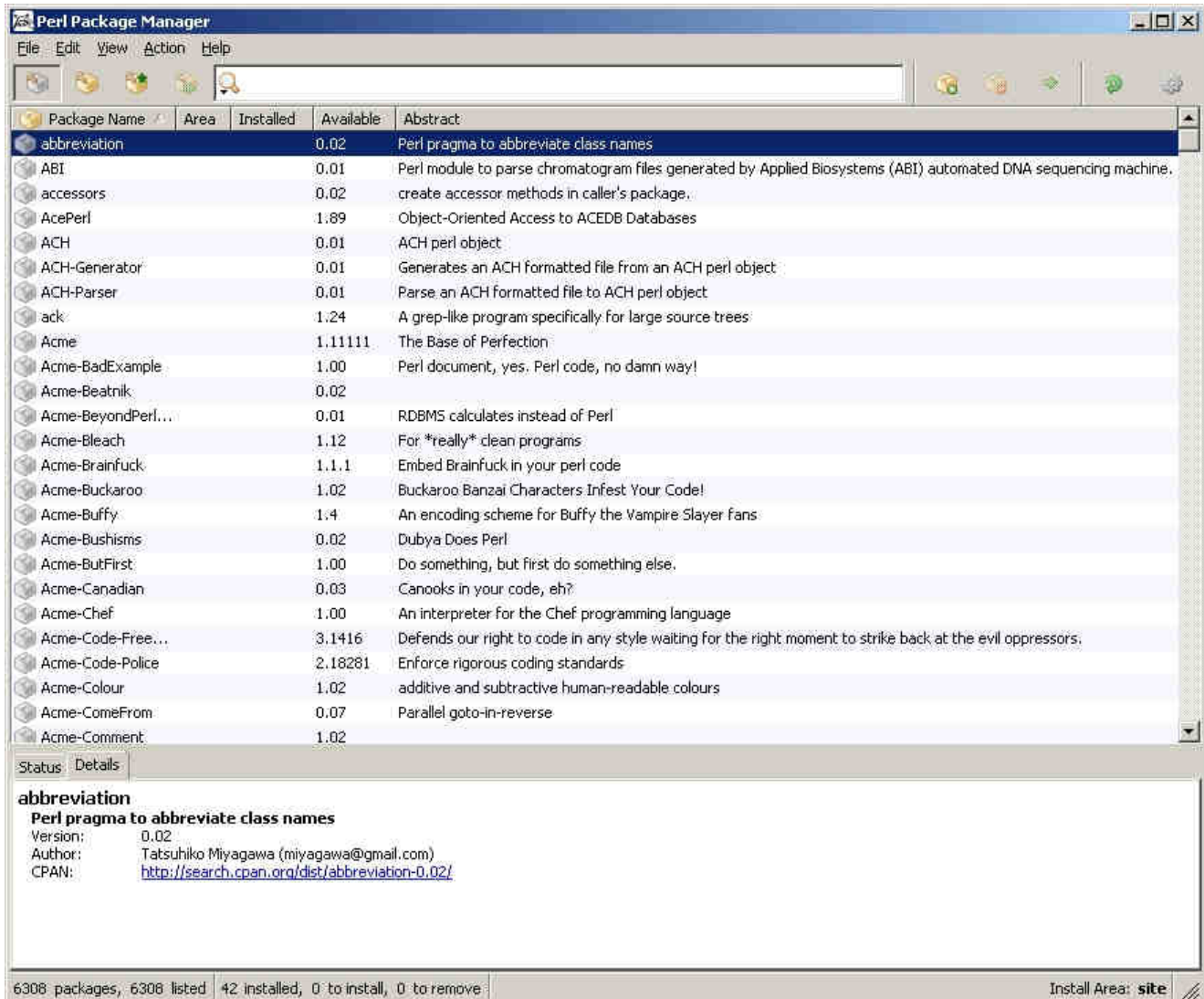
perldoc strict Donne de l'aide sur le module strict

perldoc -f print Donne de l'aide sur la fonction print

La gestion des packages

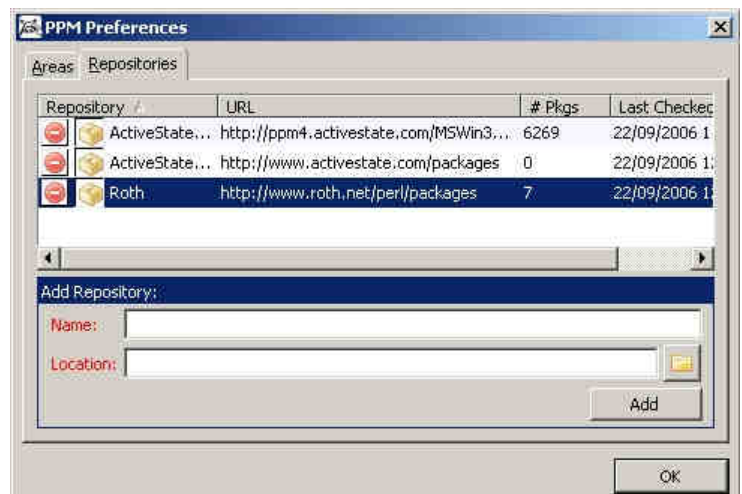
Perl Package Manager

Après l'installation de ActivePerl sous Windows, vous disposez d'un raccourci vers un outil de gestion de packages : Perl Package Manager.



Pour configurer les dépôts, allez dans Edition | Préférences. Vous pouvez ajouter les dépôts suivants :

- <http://theoryx5.uwinnipeg.ca/ppms/>
- <http://www.soulcage.net/ppds/PPDS.58/>
- <http://www.bribes.org/perl/ppm/>
- <http://trouchelle.com/ppm/>
- <http://ppm.tcool.org/archives/>



La commande Cpan

reload index recharge la liste des dépôts

Recherche de package

<http://search.cpan.org/>

Eléments du langage

Hello The World !

Forme simple

```
#!/usr/bin/perl
print "Hello the World !"
```

Forme élaborée

```
#!/usr/bin/perl
use warnings;
use strict;
my $txtMess="Hello the World\n";
print $txtMess;
```

Commentaires

Les lignes commentées sont préfixées par un #.

Exécution d'un script Perl

Sous Unix ou Linux, rendez votre script exécutable : `chmod +x hello.pl`.

Puis, lancez `./hello.pl`

Pour afficher les warnings : `perl -w hello.pl`

Pour interpréter du Perl à partir de la ligne de commande : `perl -w -e 'print "Coucou";'`

Interactivité : <STDIN>

```
#!/usr/bin/perl

print "Votre age ?";
$age=<STDIN>;
print "Vous avez $age ans.\n";
# if ($age>=18)
# {
#   print "Vous etes majeur.\n";
# }
# else
# {
#   print "Vous etes mineur.\n";
# }
print "Vous etes majeur.\n" if ($age>=18);
print "Vous etes mineur.\n" if ($age<18);
```

Les variables

Les nombres

Initialisation

Les variables numériques non initialisées ont par défaut la valeur 0.

Virgule flottante

```
$a=16.5;
$b=-9.87;
$c=.256;
$d=0.123e45;
```

Base 8 et base 16

```
$e=0756;
```

```
$f=0x1F6;
```

Les booléens

Les booléens n'existent pas en tant que type.

Est considéré comme faux toute chaîne de caractère vide ou toute valeur nulle.

Les chaînes de caractère

Initialisation

Les chaînes de caractères non initialisés sont vides.

Déclaration et concaténation

```
#!/usr/bin/perl
use warnings;
use strict;
my $texte1="mon texte\n";
#Texte interprété
my $texte2="$texte1\n";
#Texte littéral
my $texte3='$texte3\n';
my $a="a";
my $b="b";
my $c="c";
#Concaténation
print $a.$b.$c."\n";
print $texte1,$texte2,$texte3
```

Transtypage

```
#!/usr/bin/perl
use warnings;
use strict;
my $i=5;
my $j="6";
my $s=$i + $j;
print "$i+$j=$s";
```

Variable indéfinie

```
#!/usr/bin/perl
use strict;
use warnings;
#$x indéfinie
my $x;
#my $x=undef;
#$x définie
#my $x=0;
if(defined($x))
{
    print "\$x est definie"
}
else
{
    print "\$x est indefinie"
```

Séquences d'échappement

<code>\a</code>	Bip
<code>\b</code>	Backspace
<code>\cn</code>	Touche contrôle suivie du caractère n

<code>\0n</code>	Caractère en notation octale
<code>\xn</code>	Caractère en notation hexadécimale
<code>\e</code>	Touche Echap
<code>\l</code>	Lettre suivante en minuscule
<code>\L</code>	Toutes les lettres suivantes en minuscule jusqu'à <code>\E</code>
<code>\n</code>	nouvelle ligne
<code>\r</code>	retour chariot
<code>\t</code>	tabulation
<code>\u</code>	Lettre suivante en majuscule
<code>\U</code>	Toutes les lettres suivantes en majuscule jusqu'à <code>\E</code>
<code>\v</code>	Tabulation verticale
<code>\Q</code>	Interprète les méta en littéraux

Multi-affectation

```
#!/usr/bin/perl
use warnings;
use strict;
#my $i=5;
#my $j="6";
#Multiaffectation
my ($i,$j)=(5,"6");
my $s=$i + $j;
print "$i+$j=$s";
```

Les tableaux

Indice de base

L'indice de base d'un tableau est 0.

Utilisation d'un tableau

```
#!/usr/bin/perl
use strict;
use warnings;
#Initialisation d'un tableau
my @tab=(1..10,"a".."z");
#Dimension du tableau
my $dim=@tab;
#Affichage du contenu du tableau
print "@tab\n";
print "Le tableau comprend $dim elements.\n";
my $j=0;
#Lecture des éléments du tableau
for(my $i=0;$i<$dim;$i++)
{
    $j=$i + 1;
    print "Element $j = @tab[$i]\n";
}
```

Utilisation de foreach

```
#!/usr/bin/perl
use warnings;
use strict;
foreach my $ele ("a".."z")
{
    print "$ele\n";
}
ou
foreach (@tab)
```

```
{
  print "$_\n";
}
```

Test d'existence d'un élément de tableau

```
#!/usr/bin/perl
use warnings;
use strict;
my @tab=("a".."z");
foreach my $ele (@tab)
{
  print "$ele\n";
}
if(!(exists($tab[26])))
{
  print "Cet indice n'existe pas !";
}
```

Arguments de la ligne de commande

```
#!/usr/bin/perl
use warnings;
use strict;
print "$0\n";
foreach my $arg (@ARGV)
{
  print "$arg\n";
}
```

Environnement Perl

```
perl -e 'for (@INC) {printf "%d %s\n", $i++, $_}'
```

Opérateurs

Arithmétiques

+	Addition
-	Soustraction
/	Division réelle
*	Multiplication
%	Modulo
**	Puissance

```
#!/usr/bin/perl
use warnings;
use strict;
my $a=5;my $b=2;
my $r=int($a / $b);
print "La division entiere de $a par $b donne $r.\n";
$r=$a % $b;
print "Le reste de la division entiere de $a par $b donne $r.\n";
```

Concaténation

.

Le point permet de mettre bout à bout deux chaînes de caractères.

Utilisation des opérateurs arithmétiques

\$x++;	Incréméntation de \$x
\$x+=1;	Idem

Comparaison

	Numérique	Chaînes
égal	==	eq
différent	!=	ne
inférieur	<	lt
supérieur	>	gt
inférieur ou égal	<=	le
supérieur ou égal	>=	ge
comparé à	<=>	cmp

`$x<=>$y` renvoie une valeur positive si `$x` est supérieur à `$y`.

Booléens

`&&` Et
`||` Où
`!` Non

Bit à bit

`&` Et
`|` Ou
`~` Non
`^` Ou exclusif
`<<` décalage à gauche
`>>` décalage à droite

Fonctions**Arithmétiques**

`sin($x)` Sinus de `$x`
`cos($x)` Cosinus de `$x`
`exp($x)` Exponentiel de `$x`
`log($x)` Logarithme en base e de `$x`
`abs($x)` Valeur absolue de `$x`
`sqrt($x)` Racine carrée de `$x`

Chaînes

`length($x)` Retourne la longueur d'une chaîne de caractères
`chop($x)` Supprime le dernier caractère d'une chaîne
`chomp($x)` Supprime le dernier caractère d'une chaîne
`reverse($x)` Inverse la chaîne de caractères
`substr($x,$deb,$len)` Retourne une sous-chaîne à partir de `$deb` pour `$len` caractères (la chaîne démarre à 0)
`index($x,$y,$pos)` Retourne la position de la première occurrence de `$y` dans `$x` à partir de `$pos`
`rindex($x,$y,$pos)` Retourne la position de la première occurrence de `$y` dans `$x` à partir de `$pos` à partir de la droite

Tableau**Les fonctions**

`scalar(@tab)` retourne le nombre d'éléments du tableau
`unshift(@tab,exp1,exp2, ..., expN)` Ajout des éléments au début du tableau (sub)
`push(@tab, exp1,exp2, ..., expN)` Ajout des éléments à la fin Du tableau (sub)
`shift(@tab)` Retourne le premier élément du tableau et l'enlève au tableau
`pop(@tab)` Retourne le dernier élément du tableau et l'enlève au tableau
`reverse(@tab)` Inverse les éléments du tableau
`splice ???`
`join($car,@tab)` Renvoie une chaîne composée des éléments du tableau séparée par `$car`
`split(/expr. rationnelle/,chaîne)` Retourne une liste d'éléments de la chaîne déterminés par le séparateur
`grep (/expr. rationnelle/,chaîne)` Filtrage de liste

Tri d'un tableau

```
#!/usr/bin/perl
use warnings;
use diagnostics;
use strict;
my @valeurs=(5,45,12,46,89,1,542,7896,123);
#tri décroissant
@valeurs=sort({$b <=> $a} @valeurs);
print "@valeurs\n";
#tri croissant
my @chaines=("hghghg","agdgdg","zldlkd","jfhzzk","oojsjjsk","qeiuonncnc");
@chaines=sort({$a cmp $b} @chaines);
print "@chaines\n";
```

Traitement sur un tableau

```
#!/usr/bin/perl
use warnings;
use diagnostics;
use strict;
my @valeurs=(10..19);
@valeurs=map({$_*=1.1} @valeurs);
print "@valeurs\n";
```

qw

qw permet de créer un tableau à partir d'une chaîne de caractères.

```
#!/usr/bin/perl
use warnings;
use strict;
#my @tab=qw(lundi mardi mercredi jeudi vendredi samedi dimanche);
my @tab=qw/lundi mardi mercredi jeudi vendredi samedi dimanche/;
foreach (@tab)
{
    print "$_\n";
}
```

Filtrage par expression régulière

```
#!/usr/bin/perl
use strict;
use warnings;
my @liste=("lundi","mardi","mcredi","jeudi","vendredi","samedi","dimanche");
my @res=grep(/di$/,@liste);
print "@res\n";
@res=grep(/^\m/,@liste);
print "@res\n";
```

Les tables de hachage ou tableaux associatifs

L'environnement du Shell peut être lu au travers D'un tableau de hash : %ENV.

```
#!/usr/bin/perl
use strict;
use warnings;
my %liste=
(
    "lu"=>"lundi",
    "ma"=>"mardi",
    "me"=>"mcredi",
    "je"=>"jeudi",
```

```

"ve"=>"vendredi",
"sa"=>"samedi",
"di"=>"dimanche"
);
my @cles=keys(%liste);
foreach my $cle (@cles)
{
    print "Cle : $cle\tValeur : $liste{$cle}\n";
}
my @valeurs=values(%liste);
foreach my $valeur (@valeurs)
{
    print "Valeur : $valeur\n";
}
#ou
while( my ($c,$v) = each(%liste) )
{
    print "Cle : $c\tValeur : $v\n";
}

```

Affectations successives

```

my $var=20;
$var=($var++,$var - 6);

```

Structures de contrôle

Conditionnelles

Blocs

```

if(condition1)
{
    instructions1
}
elseif(condition2)
{
    instructions2
}
...
else
{
    instructions3
}

```

Modificateur d'instruction

```

instruction if(condition)
instruction if(!(condition)) est équivalent à instruction unless(condition)

```

Expression conditionnelle

```

$var=(condition)?Expr1:Expr2;

```

Répétitives

Inconditionnelles

```

for(my $compteur=0;$compteur<$max;$compteur++)
{
    instructions
}

```

Conditionnelles

```
while(condition)
{
    instructions
}
ou
do
{
    instructions
}
while(condition) ou until(!condition)
```

Next, last, redo

next Réévaluation de la condition ou incrémentation
last Fin de la boucle
redo Refait sans tester la condition ou sans incrément

Gestion des erreurs

die(message) Affiche le message et arrête le script

Les fonctions

Fonction simple

```
#!/usr/bin/perl
use warnings;
use diagnostics;
use strict;
use DateTime;
sub Age
{
    my ($texte)=@_;
    my @date=split(/\//,$texte);
    my $dt = DateTime->new
    (
        year => $date[2],
        month => $date[1],
        day => $date[0],
        hour => 0,
        minute => 0,
        second => 0,
        nanosecond => 0,
        time_zone => 'Europe/Paris',
    );
    my $now=DateTime->now( time_zone => 'Europe/Paris' );
    my $duree;
    $duree=$now-$dt;
    my $years=$duree->in_units('years');
    return $years;
}

my $saisie=<STDIN>;
print Age($saisie)."\n";
```

Récurtivité

```
#!/usr/bin/perl
use warnings;
use diagnostics;
use strict;
sub Fact
{
    my ($n)=@_;
    if($n==0)
    {
        return 1;
    }
    else
    {
        return Fact($n-1)*$n;
    }
}

print "Votre nombre ? ";
my $saisie=<STDIN>;
print Fact($saisie)."\n";
```

Valeurs de retours

Une fonction peut indifféremment retourner une liste ou un tableau.

Les fichiers

Opérateurs sur fichiers

-e	Test sur chemin valable
-f	Test si fichier normal
-d	Test si répertoire
-l	Test si lien symbolique
-r	Test si droit de lire
-w	Test si droit en écriture
-x	Test si droit en écriture
-o	Test si le fichier appartient à l'utilisateur
-z	Test si le fichier est vide
-s	Test si le fichier est non vide (sinon renvoie la taille du fichier)
-M	Renvoie l'âge du fichier en jours
-A	Age du dernier accès
-c	Nombre de jours depuis le dernier changement
-T	Test s'il s'agit d'un fichier texte
-B	Test s'il s'agit d'un fichier binaire

Test d'existence d'un fichier

```
#!/usr/bin/perl -w
use warnings;
use diagnostics;
use strict;
my $chemin="c:/boot.ini";
if (-e $chemin)
{
    print "Le fichier $chemin existe bien !";
}
```

Liste des fichiers du répertoire courant

```
#!/usr/bin/perl -w
use warnings;
use diagnostics;
use strict;
#my @fichiers=glob("c:/*.*.");
#ou
my @fichiers=<c:/*.*>;
my $taille;my $jour;
foreach (@fichiers)
{
    $taille=-s $_;
    $jour=int(-M $_);
    print "$_\t$taille\t$jour\n";
}
```

Ouverture de fichier

Aucun	Lecture
<	Lecture
>	Ecriture avec écrasement
>>	Ecriture avec ajout
+>	Lecture et écriture avec écrasement
+<	Lecture et écriture avec ajout

Lecture du contenu de fichiers texte

```
#!/usr/bin/perl -w
use warnings;
use diagnostics;
use strict;
my $chemin="c:/boot.ini";
if (-e $chemin)
{
    print "Le fichier $chemin existe bien !\n";
    open(FIC,"<$chemin") or die("Cha marche pas !");
    my $ligne;
    while($ligne=<FIC>)
    {
        chomp($ligne);
        print "$ligne\n\r";
    }
    close(FIC);
}
```

Méthode de lecture universelle

```
#!/usr/bin/perl -w
use warnings;
use diagnostics;
use strict;
my @fichiers=glob("c:/*.*ini");
my $taille=0;my $tampon="";my $lu="";
foreach (@fichiers)
{
    $taille=-s $_;
    open(FIC,"<$_");
    if($taille>0)
    {
        $lu=read(FIC,$tampon,$taille);
        print "$tampon\n";
    }
    close(FIC);
}
```

Méthode de lecture avec getc

```
#!/usr/bin/perl -w
use warnings;
use diagnostics;
use strict;
my $tampon="";my $c="";
open(FIC,"<c:/boot.ini");
while(!(eof(FIC)))
{
    $c=getc(FIC);
    $tampon.=$c;
}
print "$tampon\n";
close(FIC);
```

Fonctions

```
opendir
dirHandle
readdir
chdir
```

rewinddir
telldir
seekdir
mkdir
fileName
mode
rmdir
binmode
seek
eof
unpack
template
expr

Les expression régulières

Caractères spéciaux

\	Annule le méta
^	Début de la ligne
.	N'importe quel caractère
\$	Fin de ligne
	Alternative
()	Groupement
[]	Classe des caractères
\d	[0-9] : numérique
\D	[^0-9] : non numérique
\w	[0-9a-zA-Z_] : alpha-numérique
\W	[^0-9a-zA-Z_] : non alpha-numérique
\s	[\n\t\r\f] : espacement
\S	[^\n\t\r\f] : non espacement
\b	Limite d'un mot
\B	Autre limite
\A	Début de la chaîne
\z	Reconnaît la fin de chaîne
\Z	Reconnaît uniquement la fin de la chaîne

Cardinalités

*	0 ou plusieurs fois
+	1 ou plusieurs fois
?	0 ou 1 fois
{n}	n fois exactement
{n,}	n fois ou plus
{m,n}	au moins m fois, au plus n fois

Opérateurs

g	Recherche globale
i	Ne pas tenir compte de la casse
s	Traiter la chaîne en ligne simple
m	Traiter la chaîne en ligne multiple
o	Ne compiler l'expression une seule fois
x	Utiliser les expressions régulières étendues

Recherche et remplacement

```
#!/usr/bin/perl -w
use warnings;
use diagnostics;
use strict;
my $chaine="Cette chaine a pour vocation de tester tester les expressions regulieres. Et 1 et 2 et 3 !";
if($chaine =~ m/voc/)
{
    print "Trouve\n";
}
my $i=0;
$chaine =~ s/tester/sonner/g;
print "$chaine";
```

Variables

```
#!/usr/bin/perl -w
use warnings;
use diagnostics;
```

```
use strict;
my $chaine="Cette chaine a pour vocation de tester tester les expressions regulieres. Et 1 et 2 et 3 !";
my $i=0;
while($chaine =~ m/test/)
{
    $i++;
    print "Chaine avant :$\n";
    print "Chaine recherchee : $&\n";
    $chaine=$';
    print "$chaine\n";
}
print "occurences : $i\n";
```

Opérateur tr ou y

```
#!/usr/bin/perl -w
use warnings;
use diagnostics;
use strict;
my $chaine="Cette chaine a pour vocation de tester tester les expressions regulieres. Et 1 et 2 et 3 !";
my $texte=$chaine;
$texte=~ tr/a-z/A-Z/;
#$texte=~ y/a-z/A-Z/;
print "$texte\n";
```

Les modules

Les modules

strict Oblige à déclarer les variables
 Math::BigInt Calcul sur les grands entiers
 Math::BigFloat Calcul sur les grands nombres flottants
 Win32::Registry Utilisation du registre sous Windows

Chemins vers les modules

perl -V permet de visualiser @INC
 Pour ajouter un chemin vers des modules : set PERL5LIB=C:/Perl/lib; C:/Perl/site/lib sous Windows ou export PERL5LIB=/Perl/lib; C:/Perl/site/lib sous Linux.

Utilisation des modules courants

L'instruction use permet d'utiliser les modules.
 Les modules warnings, strict et diagnostics sont très utiles lors de l'écriture du Perl.

```
#!/usr/bin/perl
use warnings;
use diagnostics;
use strict;
use DateTime;
#use DateTime::Duration;
print "Votre age ? ";
my $saisie=<STDIN>;
my @date=split(/\//,$saisie);
my $dt = DateTime->new
(
  year => $date[2],
  month => $date[1],
  day => $date[0],
  hour => 0,
  minute => 0,
  second => 0,
  nanosecond => 0,
  time_zone => 'Europe/Paris',
);
my $now=DateTime->now( time_zone => 'Europe/Paris' );
my $duree;
$duree=$now-$dt;
my $years=$duree->in_units('years');
print "$years\n";
```

Mon premier module

Les modules ont l'extension .pm.

Création du module MesOutils.pm

```
#!/usr/bin/perl -w
package MesOutils;
use strict;
sub Saisie
{
  my ($message)=@_;
  print "$message ? ";
  my $texte=<STDIN>;
  my $len=length($texte);
  return substr($texte,0,$len - 1);
}
```

```
}
1;
```

Utilisation du module

```
#!/usr/bin/perl -w
use strict;
use MesOutils;
my $texte=MesOutils::Saisie("Votre annee");
print "$texte\n";
```

Utilisation des noms composés

Structure de répertoire

Créez le dossier MesOutils.
Renommez le module créé précédemment sous IO.
Déplacez ce module dans le dossier MesOutils.
Modifiez le contenu du module comme suit :

```
#!/usr/bin/perl -w
package MesOutils::IO;
use strict;
sub Saisie
{
    my ($message)=@_;
    print "$message ? ";
    my $texte=<STDIN>;
    my $len=length($texte);
    return substr($texte,0,$len - 1);
}
1;
```

Utilisation

```
#!/usr/bin/perl -w
use strict;
use MesOutils::IO;
my $texte=MesOutils::IO::Saisie("Votre annee");
print "$texte\n";
```

Fonctions locales aux modules

Pour déclarer une fonction interne au module :

```
#!/usr/bin/perl -w
package MesOutils::IO;
use strict;
my $fSaisie=sub
{
    my ($message)=@_;
    print "$message ? ";
    my $texte=<STDIN>;
    my $len=length($texte);
    return substr($texte,0,$len - 1);
};
sub Saisie
{
    my ($message)=@_;
    return $fSaisie->($message);
}
1;
```

Instructions par défaut

```
#!/usr/bin/perl
package outils;
use warnings;
use diagnostics;
use strict;
BEGIN
{
    print "Module outils en chargement !\n";
}
END
{
    print "Module outils en Dechargement !\n";
}
1;
```

Export des symboles***Exportation***

```
#!/usr/bin/perl -w
package MesOutils;
use warnings;
use diagnostics;
use strict;
use Exporter;
our @ISA = qw(Exporter);
our @EXPORT=qw(&Auteur);
#
#Export individuel
#
#our @EXPORT_OK=qw(&Auteur);
#
#Export par tag
#
#our %EXPORT_TAGS=(TAG=>[qw(&Auteur)]);
```

Utilisation des symboles

```
#!/usr/bin/perl -w
use strict;
use MesOutils;
#
#Utilisation des symboles exportés par EXPORT -> :DEFAULT
#Utilisation des symboles exportés par EXPORT_OK
#Utilisation des symboles exportés par EXPORT_TAGS
#
#use MesOutils qw(:DEFAULT &Auteur :TAG);
use MesOutils::IO;
Auteur();
my $texte=Saisie("Votre annee");
print "$texte\n";
```

Méthode AUTOLOAD

La méthode exportée `Essai` n'existe pas. La méthode `AUTOLOAD` intercepte l'erreur.

```
#!/usr/bin/perl -w
package MesOutils;
use warnings;
use diagnostics;
use strict;
```

```
use Exporter;
our @ISA = qw(Exporter);
our @EXPORT=qw(&Auteur &Essai);
use vars qw($AUTOLOAD);
sub Auteur
{
    print "Denis Szalkowski\n";
}
sub AUTOLOAD
{
    print "Garbage\n";
}
1;
```

Pour récupérer le nom de la fonction, utilisez la variable \$AUTOLOAD.

Documentation

```
#!/usr/bin/perl -w
package MesOutils;
use strict;
=head1 NAME
```

MesOutils.pm - ma bibliotheque d'outils

```
=head1 SYNOPSIS
```

Une bibliotheque de plus

```
=head1 DESCRIPTION
```

bah voila

```
=head2 Autres infos
```

```
=over
```

```
=item Bah oui
```

```
=item Bah non
```

```
=back
```

```
=cut
```

```
sub Auteur
{
    print "Denis Szalkowski";
}
1;
```

Appels système

Connaître la version de l'OS

`$OSNAME` ou `$$^O` Nom de la plate-forme

Récupérer la sortie d'une commande système

```
#!/usr/bin/perl -w
use strict;
system "dir";
my $sortie=`dir`;
print $sortie;
```

Utilisation de la commande open

```
#!/usr/bin/perl -w
use strict;
open DIR,"dir | ";
my @dir=<DIR>;
foreach (@dir)
{
    print "$_";
}
close DIR;
```

Exécution d'un bloc d'instructions

```
#!/usr/bin/perl -w
use strict;
system <<'BLOC';
dir
BLOC
```

Exécution et sortie de script

```
#!/usr/bin/perl -w
use strict;
exec <<'BLOC';
dir
pause
cls
BLOC
```

Commandes

`chmod` Changement des droits
`chown` Changement du propriétaire
`fnctl`
`ioctl`
`flock`
`link`
`old`
`new`
`symlink`
`readlink`

Les rapports

Champs utilisés par write

Champ	Format en sortie
@<<<	Valeur alignée à gauche
@>>>	Valeur alignée à droite
@	Valeur centrée
@##.##	Valeur numérique avec décimales fixées
@*	Chaîne de caractères sur plusieurs lignes

Champs utilisés par printf

Champ	Format	Exemple
%c	Caractère	
%s	Chaîne	%-20s : au moins 20 caractères
%d	Nombre entier	%5d : sur 5 chiffres
%x	Nombre hexa	
%o	Nombre octal	
%u	Nombre non signé	
%f	Nombre en notation normale	%5.3f : sur 3 décimales
%e	Nombre en notation scientifique	
%g	Nombre en notation compact	
%c	Caractère dont on fourni le code	
%X	Nombre en hexa avec lettres majuscules	
%E	Nombre en notation scientifique avec majuscules	

Utilisation de write

```
#!/usr/bin/perl -w
use strict;
my $valeur;my $total;
```

```
format TITRE =
Titre du rapport
```

```
.
format DETAIL =
valeur : @##.##
$valeur
```

```
.
format TOTAL =
total : @##.##
$total
```

```
.
$total=0;
my @tab=(1..10);
$~="TITRE";
write;
foreach (@tab)
{
    $valeur=$_;
    $total+=$valeur;
    $~="DETAIL";
    write;
}
$~="TOTAL";
write;
```

Variables utilisables avec write

`$%` Numéro de pages en cours
`$=` Nombre de lignes de texte par page
`$-` Nombres de lignes restantes
`$^` En-tête

Utilisation de printf

```
#!/usr/bin/perl -w
use strict;
my $valeur;my $total;
my @tab=(1..10);
$total=0;
foreach (@tab)
{
    printf("Valeur : %d\n",$_);
    $total+=$_;
}
printf("Total : %d\n",$total);
```

Introduction au module DBI de Perl

DBI (DataBase Independent) est un des modules Perl offrant aux développeurs une interface générique pour les bases de données (MySQL, PostGres, Oracle, ...). Quelle que soit la base, l'application utilise une même fonction rendant le code portable.

La table d'exemple

```
mysql> desc societe;
```

Field	Type	Null	Key	Default	Extra
id_societe	int(11)	NO	PRI	NULL	auto_increment
nom_societe	varchar(50)	YES	MUL	NULL	
cp_societe	int(11)	YES		NULL	
ville_societe	varchar(50)	YES		NULL	

Exécuter une requête SELECT

```
#!/usr/bin/perl -w
use strict;
use DBI;
#
#Liste des drivers
#
my @drivers=DBI->available_drivers;
foreach (@drivers)
{
    print "$_\n";
}
#
#Connexion
#
my $db="dbi:mysql:test";
my $login="root";
my $pwd="root";
my $conn=DBI->connect($db,$login,$pwd) or die(DBI->errstr);
#
#Requete
#
my $sql="SELECT * FROM societe";
my $req=$conn->prepare($sql);
$req->execute or die(DBI->errstr);
#
#Lecture des enregistrements
#
while(my @enr=$req->fetchrow)
{
    foreach (@enr)
    {
        print "\t$_";
    }
    print "\n";
}
#
#Nettoyage
#
$req->finish;
$conn->disconnect;
```

Saisie de données

```
#!/usr/bin/perl -w
use strict;
use DBI;
my $db="dbi:mysql:test";
my $login="root";
my $pwd="root";
my $conn=DBI->connect($db,$login,$pwd,{Autocommit=>1}) or die(DBI->errstr);
my ($nom,$cp,$ville);
my $sql;
my $choix;
do
{
    system "cls";
    print "Societe ? ";
    $nom=<STDIN>;
    chomp($nom);
    print "Cp ? ";
    $cp=<STDIN>;
    chomp($cp);
    print "Ville ? ";
    $ville=<STDIN>;
    chomp($ville);
    $sql="INSERT INTO societe(nom_societe,cp_societe,ville_societe) VALUES('$nom','$cp','$ville)";
    $conn->do($sql);
    print "Continuer (o/n) ? ";
    $choix=<STDIN>;
    chomp($cp);
}
while ($choix eq "n");
$sql="SELECT * FROM societe";
my $req=$conn->prepare($sql);
$req->execute or die(DBI->errstr);
while(my @enr=$req->fetchrow)
{
    foreach (@enr)
    {
        print "\t$_";
    }
    print "\n";
}
$req->finish;
$conn->disconnect;
```

Programmation objet

Création d'une classe

```
#!/usr/bin/perl -w
package Db;
use strict;
use DBI;

sub new
{
    my ($class,$bd,$login,$pwd)=@_;
    my $this={};
    bless($this,$class);
    $this->{conn}=DBI->connect($bd,$login,$pwd) or die("Connection echouee");
    print "Connexion reussie\n";
    return $this;
}
sub DESTROY
{
    my ($this)=@_;
    if(defined($this->{req}))
    {
        $this->{req}->finish;
    }
    $this->{conn}->disconnect;
    # $this->SUPER::DESTROY();
    print "Deconnexion reussie\n";
}
sub getConn
{
    my ($this)=@_;
    return $this->{conn};
}
sub getReq
{
    my ($this)=@_;
    return $this->{req};
}
sub execReq
{
    my ($this,$sql)=@_;
    $this->{req}=$this->{conn}->prepare($sql);
    $this->{req}->execute or die(DBI->errstr);
    print "Requete reussie\n";
}
sub listeReq
{
    my ($this)=@_;
    while(my @enr=$this->{req}->fetchrow)
    {
        foreach (@enr)
        {
            print "\t$_";
        }
        print "\n";
    }
}
```

```
1;
```

Utilisation de la classe

```
#!/usr/bin/perl -w
use strict;
use Db;
my $db=Db->new("dbi:mysql:test","root","root");
$db->execReq("SELECT * FROM societe");
$db->listeReq;
$db=undef;
```

Le Perl et le Web

Les manières d'écrire une page Html

Ecrire une page Web

```
#!/usr/bin/perl

print <<_EOT;
Content-Type: text/html

<html><head><title>Welcome to PerlWiz</title></head>
<body bgcolor=#FFCAB0>
<p><h1 style=color: red'>Welcome to the PerlWiz IDE</h1>
  <hr /style='color: darkblue'>
</p>
<p style='color: blue'>
  Click on the BROWSER OUTPUT tab to execute this program.</p>
</body></html>
_EOT
```

Script CGI

```
print "Content-type: text/html\n\n";
print "<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">","\n";
print "<html><head><title>CGI-Feedback</title></head>\n";
print "<body><h1>avis CGI du programme <i>comments.pl</i></h1>\n";
print "<p><b>nom:</b> $formulaire{utilisateur}</p>\n";
print "<p><b>commentaire:</b> $formulaire{commentaire}</p>\n";
print "</body></html>\n";
```

Utilisation du module CGI

```
#!/usr/bin/perl
use strict;
use CGI;
my $p=new CGI;
print $p->header('text/html');
print $p->start_html("test");
print $p->end_html;
```

Exécution par SSI

<pre>#exec cmd="[chemin/fichierprogramme]" cgi="[chemin-CGI/programme/script-CGI]" utilisez cmd pour exécuter un programme utilisez cgi pour exécuter un script CGI (perl, python) Les programmes doivent écrire les données sur la sortie standard. Ces sorties de données sont alors écrites directement dans le fichier HTML.</pre>	<pre><!--#exec cmd="ls" --> <!--#exec cgi="/cgi/compteur.pl" --></pre>
--	--

Gestion des formulaires

Logon.html

```
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
<title></title>
```

```

</head>
<body>
<form method="get" action="traite.pl">
  Login<br/>
  <input type="text" size="8" name="login" /><br/>
  Pwd<br/>
  <input type="password" size="8" name="pwd" /><br/>
</form>
</body>
</html>

```

logon.pl

```

#!/usr/bin/perl
use strict;
use CGI;
my $p=new CGI;
print $p->header('text/html');
print $p->start_html("logon");
print $p->start_form
(
  -method=>"get",
  -action=>"traite.pl"
);
print "Login";
print $p->br;
print $p->textfield(-name=>"login")
print $p->br;
print "Pwd";
print $p->br;
print $p->password_field(-name=>"pwd")
print $p->br;
print $p->submit();
print $p->end_form;
print $p->end_html;

```

traite.pl

```

#!/usr/bin/perl
use strict;
my $method=$ENV{'REQUEST_METHOD'};
my @tab;my $ele;my $cle;my $valeur;
print STDOUT "Content-type: text/html\n\n";
if($method eq 'GET')
{
  $buffer=$ENV{'QUERY_STRING'};
}
elsif($method eq 'POST')
{
  read(STDIN,$buffer,$ENV{'CONTENT_LENGTH'})
}
else
{
  die("Connais pas !");
}
@tab=split(/&/,$buffer);
foreach $ele (@tab)
{
  ($cle,$valeur)=split(/=/,$ele);
  $cle=~ tr/+ / /;

```

```

print "$cle : ";
$cle=s/%([a-fA-F0-9][a-fA-F0-9])/pack("C",hex($1))/eg;
$valeur=~ tr/+ / /;
$valeur=s/%([a-fA-F0-9][a-fA-F0-9])/pack("C",hex($1))/eg;
print "$valeur\n";
}

```

Redirection

```
print $p->redirect(http://www.google.fr);
```

Cookie

Création d'un cookie

```

$cook=$p->cookie(
-name=>'monCookie',
-expires=>'+24h',
-path=> '/',
-domain=>'localhost'
);
print $p->header(-cookie=>$cook);

```

Lire le cookie

```
%cook=$p->cookie(-name=>'monCookie');
```

Variables d'environnement

Vous pouvez utiliser le tableau de hachage %ENV
Mais il existe au travers du module CGI des méthodes dédiées :

accept()	Liste des types MIME
user_agent()	Identification du navigateur
remote_host()	Nom ou adresse Ip
script_name()	Nom du script
referer()	Url d'où vient la requête
server_name()	Nom du serveur
server_software()	Nom du logiciel http
remote_user()	Nom de l'utilisateur si accès protégé
request_method()	Méthode pour accéder au script
http()	Toutes les variables d'environnement

Références

Biographie

McGraw-Hill	Perl - The Complete Reference	Martin C. Brown
O'reilly	Advanced Perl Programming	
O'reilly	Perl Cookbook	
O'reilly	Perl For Oracle DBAs	
O'Reilly	Programming Web Services with Perl	
O'Reilly	Learning Perl	
O'Reilly	Perl Testing - A Developer's Notebook	

Sites

<http://perl.enstimac.fr/>

<http://www.perlfr.org/>

<http://www.comscripts.com/scripts-perl-cgi.html>

http://www.docsdunet.com/doc_per.html

<http://www.april.org/groupes/doc/perl/perl.html>

<http://stein.cshl.org/WWW/>

Documentation française

Documentation française

Exemples de scripts

Liens vers de la documentation française

Une documentation d'Alain Forcioli

le Stein Laboratory