Introduction à PERL

LPRO ISVD
Olivier FLAUZAC
olivier.flauzac@univ-reims.fr

Perl ?

Perl

- Langage de programmation créé par Larry Wall (1986)
- Créé pour simplifier le travail de scripting
- Existe sur toutes les plate-forme (Unix, linux, Windows, Mac)
- Langage gratuit
- Plus rapide que le shell
- Mise à disposition de nombreux modules

Perl : langage interprété

- Nécessite l'installation de Perl sur la machine exécutante
  - souvent installé lors de l'installation du système
- Gestion de la mémoire laissée à l'interpréteur
- Portabilité du code

Perl : Utilisation

- Manipulation du système de fichier et des fichiers
- Manipulation des processus
- Gestion des commandes externes
- Gestion des textes
- Gestion des expressions régulières

Premier exemple

- Définition de l'environnement
- Enregistrement dans un fichier texte
- Les instructions se terminent par ;
- Modification des droits du fichier (exécutable)
- Fichier à l'extension .pl

```
#!/usr/bin/perl
# un commentaire
print "Salut les amis \n";
```

Les données en perl

Variables

- Eléments de stockage
- «Typage» entre les
  - ☞ variables scalaires $nom_variable
  - ☞ tableaux / listes @nom_tableau
  - ☞ tableaux associatif (hash) %nom_hash
- Variables «mixtes»
  - ☞ entiers
  - ☞ chaînes de caractères
  - ☞ descripteurs de fichiers ...

Variables numériques

- Variables entières et à virgule
- Opérations
  - ☞ affectation : =
  - ☞ comparaison : == ; < ; <= ; > ; >= ; !=
  - ☞ opérations : + ; - ; / ; *

Variables numériques exemple

```
#!/usr/bin/perl
$a = 15;
print "la valeur de a est : $a \n";
$b = 75;
$c = $a + $b;
print "la valeur de a : $a , de b : $b ";
print " et de c : $c \n";
```

Variables booléennes

- Représentation des valeurs binaires
  ☞ Vrai ou Faux

- Opérations
  ☞ affectation : =

  ☞opérateurs : && ; ||

- Résultats des opérations de comparaison

Variables booléennes exemple

```
#!/usr/bin/perl
$a = true;
$b = false;

print "Valeur de a : $a , valeur de b : $b \n";
$c = $a && $b;
$d = $a || $b;

print "Valeur de c : $c , valeur de d : $d \n";
```

Variables chaînes de caractères

- Utilisation du caractère $
- Définition entre les guillemets
- Opérateurs de comparaison : gt , ge , lt , le , eq , ne

Variables chaînes de caractères exemple

```
#!/usr/bin/perl
$ch1 = "salut";
$ch2 = "les";
$ch3 = "copains";

$ch4 ="$ch1 $ch2 $ch3";

print "$ch4 \n";
print "$ch1 $ch2 $ch3";
```

Tableaux

- Série indexée de valeurs
- Index débute à 0
- Structures Dynamiques
- Tableau désigné par le caractère @
- Case d'un tableau désignée par la $
- Index de la dernière case : $#

Tableaux exemple

```
#!/usr/bin/perl
@tab=("toto","titi",3,45);
print("première case : $tab[0] \n");
print("seconde case : $tab[1] \n");

$derniere = $#tab;
print ("indice de la dernière case :
$derniere \n");
$suiv = $derniere + 1;
$tab[$suiv]="une case de plus";
print("la dernière case : $tab[$#tab] \n");
```

Tableaux associatifs
Tables de Hachage

- Tableau dont les clés sont des chaînes
- Association chaine / valeur
- Permet de définir des tableaux de propriétés
- Désigné par le caractère %
- Un élément désigné par le caractère $

Exemple

```
#!/opt/local/bin/perl

%urls= (
"google"=>"http://www.google.fr",
"yahoo"=>"http://www.yahoo.fr",
"altavista"=>"http://fr.altavista.com"
);

print "L'adresse de google est : ".
      $urls{"google"}.
      "\n";
```

Variables particulières

- @ARGV : tableau des paramètres du script
- $_ : variable par défaut
  ☞ dernière lecture

  ☞ dernière exécution

- $!
  ☞ dernière erreur

- %ENV
  ☞ variables d'environnement

Structures de contrôle

Structures de contrôle

- Gestion de l'exécution
- Structures alternatives
  - ☞ if, if ... else ...
  - ☞ unless
- Structures itératives
  - ☞ while
  - ☞ for
  - ☞ foreach

Structures de contrôle (IF et IF ... ELSE...)

```
#!/usr/bin/perl
$v1 = 12;
if($v1 < 15){
   print("v1 est inférieur à 15 \n");
}
$v2 = 18;
if($v2 < 15){
   print("v2 est inférieur à 15 \n");
}else{
   print("v2 est supérieur ou égal à 15 \n");
}
```

Structures de contrôle (unless)

```
#!/opt/local/bin/perl

$a = 15;
unless($a == 15){
  print "a n'est pas égal à 25";
}else{
  print "a est égal à 25";
}
```

Structures de contrôle (WHILE)

```
#!/usr/bin/perl

@tab=(7,8,99,3,45);
$i = 0;

while($i <= $#tab){
    print("case $i $tab[$i] \n");
    $i++; # comme $i = $i + 1
}
```

Structures de contrôle (WHILE)

```
#!/usr/bin/perl
@tab=(7,8,99,3,45);
$i = 0;
while($i <= $#tab && $tab[$i] != 3){
   $i++; # comme $i = $i + 1
}

if($i <= $#tab){
   print("3 est dans le tableau \n");
}else{
   print("3 n'est pas dans le tableau \n");
}
```

Structures de contrôle (FOR - FOREACH)

```
#!/usr/bin/perl

@tab=(7,8,99,3,45);

for($i=0;$i<=$#tab;$i++){
  print("case $i : $tab[$i] \n");
}

foreach $val (@tab){
  print ("valeur : $val \n");
}
```

Parcours d'un table de hachage

```
#!/usr/bin/perl

print "Liste des informations de votre système :\n";

while (($cle, $valeur) = each (%ENV)){
        print "$cle a comme valeur $valeur.\n";
}

print "--------------------------\n";

foreach $val (keys %ENV){
   print "$val \n";
}
```

Flux et fichiers

Notion de flux

- Source de données
- Flux standards :
  ☞ STDIN , STDOUT , STDERR
- Flux associés aux fichiers textes

Affichage et flux

- print STDOUT «chaîne de caractère»;
- print STDERR «chaîne de caractères»

```
#!/usr/bin/perl
print STDOUT "Bonjour à tous \n";
print STDERR "Salut les amis \n";
```

Lecture dans le flux clavier

- Utilisation du flux d'entrée standard
- Stockage de la lecture dans une variable
- Lecture avec <STDIN>

```
#!/usr/bin/perl
print "Entrez au clavier :";
$a=<STDIN>;
print "vous avez entré : $a \n";
```

Utilisation des fichiers

- Gestion des fichiers :
  - ☞ gestion du contenu
    - ☞ lecture des données
    - ☞ écriture des données
  - ☞ gestion du système de fichier
- Gestion des systèmes de fichier
  - ☞ gestion des répertoires
  - ☞ gestion des fichiers

Accès au contenu des fichiers

1. ouverture du fichier
   - ☞ open(descripteur, "(mode)nomfichier");
   - ☞ modes : **<** lecture ; **>** écriture ; **>>** ajout
2. lecture ou écriture
   - ☞ lecture : <descripteur>
   - ☞ écriture : print descripteur ".....";
3. fermeture du fichier
   - ☞ close descripteur

Exemple d'utilisation

```
#!/usr/bin/perl
open(FIC,">fichier.txt");
print FIC "une première ligne \n";
print FIC "une seconde ligne \n";
print FIC "une troisième ligne \n";
print FIC "une quatrième ligne \n";
close FIC;
```

Exemple d'utilisation

```
#!/usr/bin/perl
open(FIC,"<fichier.txt");
while(<FIC>){
   print $_;
}
close FIC;
```

```
#!/usr/bin/perl
open(FIC,"<fichier.txt");
while($a = <FIC>){
   print $a;
}
close FIC;
```

Gestion des erreurs

- Utilisation de die
- Utilisation combinée avec les opérations

```
#!/usr/bin/perl
open(FIC,"<fichier.txt") || die "Erreur ouverture de fichier";
while(<FIC>){
   print $_;
}
close FIC || die "Erreur fermeture de fichier";
```

Le système de fichier

- Opérations de tests sur les fichier
- Opération de manipulation
  - ☞ fichiers
  - ☞ répertoires

Information sur les fichiers

-e	test de l'existence du fichier / répertoire
-r	droit de lecture
-w	droit d'écriture
-x	droit d'exécution
-d	test de répertoire
-f	test de fichier
-z	test de taille non nulle

Opération sur les fichiers

chmod	modification des droits
mkdir	création de fichier
rename	renommage de fichier
rmdir	suppression d'un répertoire
chdir	changement de répertoire de travail
opendir	ouverture d'un répertoire
readdir	lecture d'une entrée du répertoire
closedir	dermeture du répertoire

Opération sur les fichiers exemple

```
#!/usr/bin/perl
opendir(REP, ".") or die "impossible d'ouvrir le répertoire";
@fichiers = readdir REP;
closedir REP;
foreach $entree (@fichiers){
   if(-f $entree){
      print "$entree est un fichier \n";
   }else{
      print "$entree est un répertoire \n";
   }
}
```

FONCTIONS

Gestion horaire

```
#!/usr/bin/perl
# time temps UNIX
$t = time();
print "temps UNIX : $t \n";
$l = localtime();
print "temps local : $l \n";
($sec, $min, $h, $j, $m, $a, $sj, $aj, $isdst) = localtime();
$annee = 1900 + $a;
$mois = $m+1;
print "nous somme le $j / $mois / $annee - il est $h : $min \n";
```

Manipulation des chaînes

- Longueur d'une chaîne : length($ch)
- Conversion minuscule : lc($ch)
- Conversion majuscule : uc($ch)
- Suppression du dernier caractère : chop($ch)
- Suppression du retour chariot à la fin de la chaîne : chomp($ch)
- Concaténation : utilisation du caractère .

Manipulation des tableaux

- suppression et retour du premier élément : shift(@tab)
- ajout d'un élément au début du tableau : unshift(@tab,$valeur);
- ajout d'un élément à la fin du tableau : push(@tab,$valeur)
- suppression etr retour du dernier élément : pop(@tab)

Fonctions système

- exécution d'un programme
  ☞ exec ("commande");
- Suspension du script, sortie à l'écran
  ☞ system("commande");
- Apostrophes inversées , récupération du résultat
  ☞ `commande`;
- fin du script
  ☞ exit

Fonctions système

```
#!/usr/bin/perl
system("tar -cvf fichiers.tar .");

$result = `ls -al`;
print "résultat de l'exécution : \n
$result";
```

Fonctions utilisateur

- Définition de fonctions utilisateur
- Utilisation du mot clé `sub`
- Pas de définition explicite des paramètres
- Liste des paramètres : `@_`

Définition des fonctions

```
#!/usr/bin/perl

sub fct1{
    print "Bonjour les amis \n";
}

sub fct2{
    foreach (@_){
        print "la valeur $_ \n";
    }
}
```

```
sub fct3{
    my $res;
    if($_[0] < $_[1]){
        $res = $_[0];
    }else{
        $res = $_[1];
    }
    return $res;
}

sub fct4{
    $res = 0;
    foreach (@_){
        $res = $res + $_;
    }
    return $res;
}
```

Appels des fonctions

```
print"********************\n";
print "appel de fct1 \n";
fct1;

print"********************\n";
print "appel de fct2 \n";
fct2("salut","hello",3,78);

print"********************\n";
print "appel de fct3 \n";
$val = fct3(6,8);
print "Minimum de 6,8 : $val \n";

print"********************\n";
print "appel de fct4 \n";
$val = fct4(6,8,6,8,9,10);
print "Somme des valeurs : $val \n";
```

Expressions régulières

Expressions régulières

- Utilisation pour :
  - de la recherche de motif
  - de la substitution de motif
- Utilisation de
  - =~ pour la vérification
  - !~ pour le contraire
  - application de l'expression de droite à la chaîne de gauche

Recherche de motif

- Utilisation pour de m (*member*)
- option :
  - Pas de prise en compte de la casse i (ignore)
- m/expression/options

Recherche de motif exemple

```
#!/usr/bin/perl
@tab=("olivier","oreste","jean","gaston","oscar");

foreach $val (@tab){
   if($val =~ m/^o/){ print "$val \n";}
}

print ("**************************\n");

foreach $val (@tab){
   if($val !~ m/^o/){ print "$val \n"; }
}
```

Substitution

- Utilisation pour de s (substitute)
- options :
  - Pas de prise en compte de la casse i (ignore)
  - Substitution globale g
- s/expression/remplacement/options

Substitution de motif exemple

```
#!/usr/bin/perl
$chaine="bonjour les amis";
$chaine =~ s/amis/copains/;
print "$chaine \n";

$chaine2="Nous somme le 16 Novembre 2012";
$chaine2=~ s/ Novembre /-11-/;
print "$chaine2 \n";

$chaine3="    Salut      les     amis      ";
$chaine3=~ s/^ +//;
$chaine3=~ s/ +$//;
$chaine3=~ s/ +/ /g;
print "-->$chaine3<--\n";
```