Perl pour physiciens

Nicolas Regnault

Laboratoire Pierre Aigrain

19 novembre 2004

Plan du cours :

- introduction
- les variables
- les tableaux
- les fichiers
- les expressions régulières
- les fonctions
- hygiène et programmation (les 10 commandements)
- PERL maître du monde
- Exemples à la demande

Plan du cours :

- introduction
- les variables
- les tableaux
- les fichiers
- les expressions régulières
- les fonctions
- hygiène et programmation (les 10 commandements)
- PERL maître du monde
- Exemples à la demande

Chapitre 3 :
Les tableaux

- alloués dynamiquement
- premier indice = 0
- pas de type fixé pour l'intégralité du tableau
- nom doit commencer par @

particule.pl

```
#!/usr/bin/perl

@particule = ("photon", "boson", 0, 0, 1);
print ("nom : ".$particule[0]."\n");
print ("statistique : ".$particule[1]."\n");
print ("masse : ".$particule[2]." eV\n");
print ("charge : ".$particule[3]."\n");
print ("spin :".$particule[4]."\n");
```

particule.pl

```
#!/usr/bin/perl

@particule = ("photon", "boson", 0, 0, 1);
print ("nom : ".$particule[0]."\n");
print ("statistique : ".$particule[1]."\n");
print ("masse : ".$particule[2]." eV\n");
print ("charge : ".$particule[3]."\n");
print ("spin :".$particule[4]."\n");
```

particule.pl

```
#!/usr/bin/perl

@particule = ("photon", "boson", 0, 0, 1);
print ("nom : ".$particule[0]."\n");
print ("statistique : ".$particule[1]."\n");
print ("masse : ".$particule[2]." eV\n");
print ("charge : ".$particule[3]."\n");
print ("spin :".$particule[4]."\n");
```

affecter un élément au tableau

```
$particule[0] = "photon";
$particule[1] = "bosons";
```

affecter un élément au tableau

```
push (particule, "photon") ;
push (particule, "boson") ;
```

parcours.pl

```
#!/usr/bin/perl

@energies = (1, 4, 9, 16, 25, 36, 49, 64);
$i = 0;
while ($i < 8)
{
print ("energie ".$i." : ".$energies[$i]."\n");
$i++;
}
```

parcours.pl

```
#!/usr/bin/perl

@energies = (1, 4, 9, 16, 25, 36, 49, 64);
$i = 0;
while ($i < 8)
{
print ("energie ".$i." : ".$energies[$i]."\n");
$i++;
}
```

parcours.pl

```
#!/usr/bin/perl

@energies = (1, 4, 9, 16, 25, 36, 49, 64);
$i = 0;
while ($i < $#energies)
{
print ("energie ".$i." : ".$energies[$i]."\n");
$i++;
}
```

parcours.pl

```
#!/usr/bin/perl

@energies = (1, 4, 9, 16, 25, 36, 49, 64);
$i = 0;
while ($i < $#energies)
{
print ("energie ".$i." : ".$energies[$i]."\n");
$i++;
}
```

foreach.pl

```
#!/usr/bin/perl

@energies = (1, 4, 9, 16, 25, 36, 49, 64);
foreach $valeur (@energies)
{
print ($valeur."\n");
}
```

Pas de foreach dans un foreach !

foreach.pl

```
#!/usr/bin/perl

@energies = (1, 4, 9, 16, 25, 36, 49, 64);
foreach $valeur (@energies)
{
print ($valeur."\n");
}
```

Pas de foreach dans un foreach !

foreach.pl

```
#!/usr/bin/perl

@energies = (1, 4, 9, 16, 25, 36, 49, 64);
foreach $valeur (@energies)
{
print ($valeur."\n");
}
```

Pas de foreach dans un foreach !

sort.pl

```
#!/usr/bin/perl

@particules = ("photon","electron","proton","neutron","muon","pion");
@particulestriees = sort (@particules);
foreach $tmp (@particulestriees)
{
print ($tmp."\n");
}
```

tri_numerique.pl

```
#!/usr/bin/perl

@masses = (938.3, 939.6, 0.511);
@massestriees = sort {$a <=> $b} (@masses);
  foreach $tmp (@massestriees)
{
print ($tmp."\n");
}
```

tri_numerique.pl

```
#!/usr/bin/perl

@masses = (938.3, 939.6, 0.511);
@massestriees = sort {$a <=> $b} (@masses);
  foreach $tmp (@massestriees)
{
print ($tmp."\n");
}
```

tri_numerique.pl

```
#!/usr/bin/perl

@masses = (938.3, 939.6, 0.511);
@massestriees = sort {$b <=> $a} (@masses);
 foreach $tmp (@massestriees)
{
print ($tmp."\n");
}
```

argv.pl

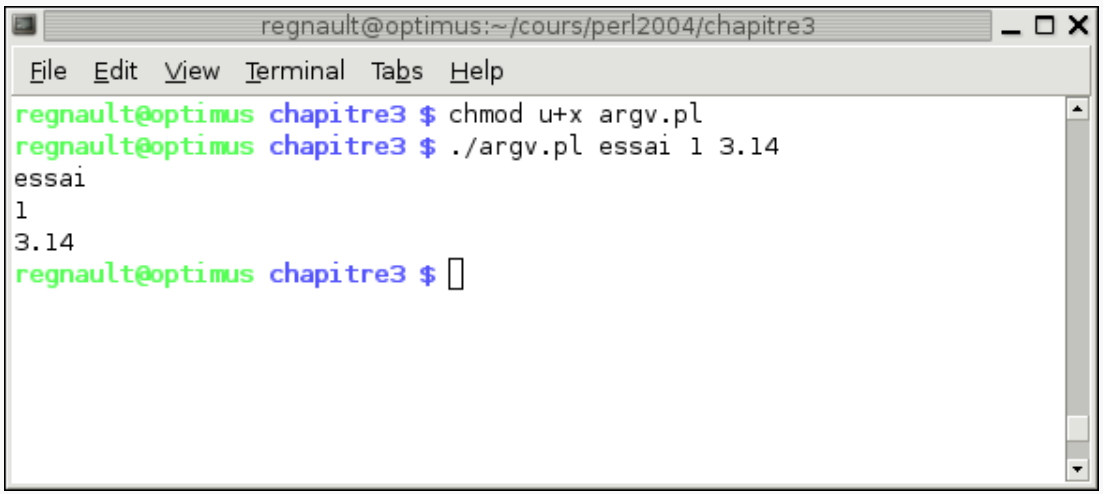
```
#!/usr/bin/perl

foreach $argument (@ARGV)
{
print ($argument."\n");
}
```

argv.pl

```
#!/usr/bin/perl

foreach $argument (@ARGV)
{
print ($argument."\n");
}
```


regnault@optimus:~/cours/perl2004/chapitre3
File Edit View Terminal Tabs Help
regnault@optimus chapitre3 $ chmod u+x argv.pl
regnault@optimus chapitre3 $ ./argv.pl essai 1 3.14
essai
1
3.14
regnault@optimus chapitre3 $

- Créez un tableau de nombres aléatoires dont la taille sera passée comme premier argument.
- Cherchez le minimum et le maximum par deux méthodes
  - rapide à exécuter
  - lente mais compacte.

Chapitre 4 :
Les fichiers

tests.pl

```
#!/usr/bin/perl

$fichier = "toto";
if (-e $fichier)
{
print ("le fichier ".$fichier." existe\n");
}
```

tests.pl

```
#!/usr/bin/perl

$fichier = "toto";
if (-e $fichier)
{
print ("le fichier ".$fichier." existe\n");
}
```

tests.pl

```
#!/usr/bin/perl

$fichier = "toto";
if (-d $fichier)
{
print ("le fichier ".$fichier." est un répertoire\n");
}
```

tests.pl

```
#!/usr/bin/perl

$fichier = "toto";
if (-d $fichier)
{
print ("le fichier ".$fichier." est un répertoire\n");
}
```

tests.pl

```
#!/usr/bin/perl

$fichier = "toto";
if (-f $fichier)
{
print ("le fichier ".$fichier." est un fichier\n");
}
```

tests.pl

```
#!/usr/bin/perl

$fichier = "toto";
if (-f $fichier)
{
print ("le fichier ".$fichier." est un fichier\n");
}
```

- *-e* teste si le fichier existe.
- *-d* teste si le fichier est un répertoire.
- *-f* teste si le fichier est bien un fichier et non un répertoire.
- *-r* teste si le fichier est accessible en lecture.
- *-w* teste si le fichier est accessible en écriture.
- *-x* teste si le fichier est exécutable.

tests.pl

```
#!/usr/bin/perl

$fichier = "toto";
if ((-f $fichier) && (-x $fichier))
{
print ($fichier." est exécutable\n");
}
```

tests.pl

```
#!/usr/bin/perl

$fichier = "toto";
if ((-f $fichier) && (-x $fichier))
{
print ($fichier." est exécutable\n");
}
```

lecture.pl

```
#!/usr/bin/perl

open (INFILE, "test.txt");
while ($ligne = <INFILE>)
{
print ($ligne);
}
close (INFILE);
```

entree.pl

```
#!/usr/bin/perl

$a = <STDIN>;
print ("vous venez de taper ".$a);
```

entree.pl

```
#!/usr/bin/perl

$a = <STDIN>;
print ("vous venez de taper ".$a);
```

lecture.pl

```
#!/usr/bin/perl

unless (open (INFILE, "test.txt"))
{
die ("impossible de lire le fichier test.txt\n");
}
while ($ligne = <INFILE>)
{
print ($ligne);
}
close (INFILE);
```

lecture.pl

```
#!/usr/bin/perl

unless (open (INFILE, "test.txt"))
{
die ("impossible de lire le fichier test.txt\n");
}
while ($ligne = <INFILE>)
{
print ($ligne);
}
close (INFILE);
```

lecture.pl

```
#!/usr/bin/perl

if ( !(open (INFILE, "test.txt")))
{
die ("impossible de lire le fichier test.txt\n");
}
while ($ligne = <INFILE>)
{
print ($ligne);
}
close (INFILE);
```

lecture.pl

```
#!/usr/bin/perl

unless (open (INFILE, "test.txt"))
{
die ("impossible de lire le fichier test.txt\n");
}
while ($ligne = <INFILE>)
{
print ($ligne);
}
close (INFILE);
```

lecture.pl

```
#!/usr/bin/perl

{
 die ("impossible de lire le fichier test.txt\n");
}
while ($ligne = <INFILE>)
{
print ($ligne);
}
close (INFILE);
```

ecriture.pl

```
unless (open (OUTFILE, ">factoriel"))
  {
die ("impossible de creer le fichier factoriel\n");
}
$fact = 1;
for ($iter = 1; $iter <= 10; $iter++)
{
$fact *= $iter;
print OUTFILE ($fact."\n");
}
close (OUTFILE);
```

ecriture.pl

```
unless (open (OUTFILE, ">factoriel"))
  {
die ("impossible de creer le fichier factoriel\n");
}
$fact = 1;
for ($iter = 1; $iter <= 10; $iter++)
{
$fact *= $iter;
print OUTFILE ($fact."\n");
}
close (OUTFILE);
```

ecriture.pl

```
unless (open (OUTFILE, ">>factoriel"))
 {
die ("impossible de creer le fichier factoriel\n");
}
$fact = 1;
for ($iter = 1; $iter <= 10; $iter++)
{
$fact *= $iter;
print OUTFILE ($fact."\n");
}
close (OUTFILE);
```

ecriture.pl

```
unless (open (OUTFILE, ">factoriel"))
  {
die ("impossible de creer le fichier factoriel\n");
}
$fact = 1;
for ($iter = 1; $iter <= 10; $iter++)
{
$fact *= $iter;
print OUTFILE ($fact."\n");
}
close (OUTFILE);
```

ls.pl

```
#!/usr/bin/perl

foreach $fichier (<*>)
  {
print ($fichier."\n");
}
```

ls.pl

```
#!/usr/bin/perl

foreach $fichier (<*>)
  {
print ($fichier."\n");
}
```

ls.pl

```
#!/usr/bin/perl

foreach $fichier (<*.pl>)
 {
print ($fichier."\n");
}
```

- *rename* renommer un fichier *rename(ancien_nom, nouveau_nom)*.
- *unlink* effacer un fichier.
- *mkdir* créer un répertoire.
- *rmdir* effacer un répertoire.

- créez cent fichiers **testxx** (avec **xx** allant de 0 à 99) (chaque fichier contenant son propre nom).
- renommez ces fichiers de **testxx** en **totoyy** avec **yy = xx+100**.

- créez un fichier **donnees1.dat**, chaque ligne est un nombre de 0 à 1 avec un pas de 0.01
- créez un fichier **donnees2.dat** à partir de **donnees1.dat** avec à chaque ligne $x\ cos(2\pi x)$