

Premier support (plus que lapidaire) du cours PERL

Pierre Pompidor,
LIRMM, 161 rue Ada,
34392 Montpellier Cedex 5
tel: 67 41 85 36, fax: 67 41 85 00
email: pompidor@lirmm.fr

12 octobre 1999

Table des matières

1 Introduction :	1
2 Deux fonctions prédéfinies de base :	1
3 Un exemple, juste pour avoir un aperçu de la chose :	2
4 Les variables :	2
4.1 Les variables individuelles (ou scalaires) :	2
4.2 Les tableaux simples (préfixés par @) :	2
4.3 Les tableaux associatifs (préfixés par %) :	2
5 Les structures conditionnelles et itératives :	3
5.1 Les structures conditionnelles :	3
5.2 Les structures itératives :	3
5.3 Les commandes de contrôle des structures itératives :	3
6 Les opérateurs de comparaison :	4
7 Les expressions régulières :	4

1 Introduction :

Synthèse hardie de C, d'interpréteurs de commandes SHELL et d'utilitaires UNIX (sed, awk ...).
Langage interprété, difficile, utilisé pour l'écriture de **scripts**.

Différents modes de lancement d'un script (il doit être exécutable) :

La première ligne du script doit être : **#!/usr/bin/perl** ou **#!/usr/local/bin/perl**

directement :	script_perl
directement avec paramètres :	script_perl paramètre_1 paramètre_2 ...
avec en entrée le contenu d'un fichier :	cat fichier script_perl
avec en entrée le contenu d'un fichier et des paramètres :	cat fichier script_perl paramètre_1 paramètre_2 ...

Les lignes (ou fin de lignes) en commentaires sont précédées par #

Référence principale : Programmation en PERL, deuxième édition en français, Larry Wall, O'Reilly

2 Deux fonctions prédéfinies de base :

affichage sur la sortie standard	print liste_de_valeurs_à_afficher	print "résultat", \$resultat, "\n"
découpage suivant un motif	split /motif de séparation/, expression	@champs = split /:/, \$ligne

5 Les structures conditionnelles et itératives :

5.1 Les structures conditionnelles :

```
if ( expression conditionnelle )
{ ... }
else
{ ... }
```

Les accolades sont toujours obligatoires

SWITCH :

```
{
    if ( ... ) { ...; last SWITCH; }
    if ( ... ) { ...; last SWITCH; }
    ...
}
```

Pas d'équivalence du switch C/JAVA

*Les labels (ici SWITCH) et la commande **last** sont présentés plus loin*

5.2 Les structures itératives :

```
while (expression conditionnelle)
{ ... }
```

Recherche d'un motif sur les lignes d'un fichier passé sur l'entrée standard :

```
while (<STDIN>)
{
    if (/expression régulière/) {...}
}
```

```
for (initialisation de l'indice; expression conditionnelle; incrémentation/décrémentation)
{ ... }
```

```
for (i=0; i < 10; i++)
{ ... }
```

```
foreach $variable (liste de valeurs)
{ ... }
```

Traitement sur une liste d'utilisateurs :

```
foreach $user (@users)
{...}
```

5.3 Les commandes de contrôle des structures itératives :

LABEL_EN_MAJUSCULES :	pose une marque (un label)
goto	permet de sauter au label concerné
last	sortie immédiate de la boucle
next	saut immédiat à la prochaine itération
redo	saut immédiat à la prochaine itération sans évaluer l'expression conditionnelle

6 Les opérateurs de comparaison :

nombres	chaînes	
==	eq	vrai si le premier opérande est égal au second
!=	ne	vrai si le premier opérande n'est pas égal au second
<	lt	vrai si le premier opérande est inférieur au second
>	gt	vrai si le premier opérande est supérieur au second
<=	le	vrai si le premier opérande n'est pas le plus grand que le second
<=>	cmp	0 si égal, 1 si le premier opérande plus grand, -1 si c'est le second

Un type d'expression spécifique permet de tester le statut d'un fichier.

Ce type d'expression a la forme générale : **-spécification référence** :

d type répertoire, **f** type ordinaire (fichier régulier), **T** type fichier texte, **e** existence du fichier, **r** droit de lecture, **w** droit d'écriture, **x** droit d'exécution

7 Les expressions régulières :

Une expression régulière définit une chaîne de caractère quelconque.

Les expressions régulières sont encadrées par des / La variable \$& va contenir la chaîne de caractères appariée.

- n'importe quel caractère
- | exprime l'alternative
pierre|paul|jacques ↔ pierre ou paul ou jacques
- () groupement de caractères
effet secondaire : le groupement de caractères est mémorisé dans une variable nommée de \$1 à \$9
- (? :) supprime la mémorisation dans une variable
- (? =) les caractères appariés ne sont pas mémorisés dans \$&
- [] constitution d'un ensemble de caractères dans lequel un caractère pourra être choisi
[a-zA-Z] : un des caractères alphabétiques
- ^ appariement en début de ligne
- \$ appariement en fin de ligne
- * répétition de 0 à n fois du caractère (ou du groupement de caractères) précédent
- + répétition de 1 à n fois du caractère (ou du groupement de caractères) précédent
- ? répétition de 0 à 1 fois du caractère (ou du groupement de caractères) précédent
- {n} répétition n fois du caractère (ou du groupement de caractères) précédent
- {n,} répétition au moins n fois du caractère (ou du groupement de caractères) précédent
- {n,m} répétition de n à m fois du caractère (ou du groupement de caractères) précédent
moo{3} ↔ moooo
(moo){3} ↔ moomoomoo
- \w un caractère alphanumérique, soit un élément de l'ensemble suivant : [a-zA-Z_0-9]
- \w+ un mot composé des caractères alphanumériques précédents
- \W un caractère non alphanumérique
- \b une frontière de mot (entre \w et \W)
- \s un élément de l'ensemble suivant : [\t\n\r\f]
- \S tout élément non compris dans l'ensemble précédent

Remplacement de chaînes de caractères :

s/chaîne à remplacer/chaîne de remplacement/options

Options possibles :

- g remplace toutes les occurrences
- i ne distingue pas les minuscules des majuscules

Exemples :

- s/^([^]) + ([^])/\$2 \$1/ inversion des deux premiers mots
- s/\bmot1\b/mot2/g remplacement de tous les "mot1" par "mot2"
- \$nombre =~ s/\bmot1\b/mot2/g remplacement et comptage du nombre de remplacement