Support de cours n°3

DESS TEXTE

Introduction

Ce support de cours introduit brièvement la notion de modules en langage Perl. Il fournit les bases pour l'écriture d'un module, et surtout présente comment appeler des fonctions de modules existants.

1 Définition d'un module

- Un module est un fichier perl regroupant un ensemble de variables et/ou de fonctions qui vont être utilisées par un programme *client*.

- Le nom du fichier contenant la définition d'un module doit avoir pour nom le nom du module suivi de l'extension **.pm**.

- Enfin ce fichier doit être placé dans un répertoire présent dans le tableau prédéfini **@INC**. Ce tableau contient le répertoire de travail (noté "."). Au besoin dans votre programme *client*, vous pouvez ajouter en tête de ce tableau le répertoire où vous stockez vos modules via `unshift(@INC,"c :\\Modules") ;`

- Le contenu du fichier `Monmodule.pm` intègre les éléments suivants :

```
package Monmodule;  #déclaration du module
sub toto {
...
}
sub tata {
...
}
1;
```

  Ce module contient les fonctions `toto` et `tata`. Il débute par l'instruction de déclaration de module, et finit par `1 ;` signifiant au programme *client* que le chargement du module s'est déroulé normalement.

- Il faut noter que, par convention, les noms de modules commencent par une majuscule, suivie de minuscules.

2 Utilisation d'un module dans un fichier *client*

Voici comment appeler les fonctions du module `Monmodule.pm` :

```
#!/usr/bin/perl
use strict;
use Monmodule;
Monmondule::toto("arg1","arg2"); #en imaginant que toto a deux arguments
```

A quoi servent les " : :" ?

Réponse : à éviter les collisions de noms (par exemple si vous définissez une autre fonction `toto` dans votre programme perl).

3 Classement des modules

- il est possible de classer les modules dans des répertoires, par exemple : un répertoire `MesModules` contenant tous vos modules pesonnels (et dont `Monmodule.pm`).
  L'appel aux fonctions de `Monmodule.pm` se fera alors via :
  `MesModules::Monmodule::toto("arg1","arg2");`

4 Documentation d'un module

On documente un module en plaçant dans son code source (fichier d'extension **.pm**) des commentaires saisis suivant une syntaxe particulière : la syntaxe POD.
Exemple :

```
=head1 NAME
Monmodule - module contenant les fonctions toto et tata
=head1 SYNOPSIS
Monmodule::toto("arg1","arg2");
Monmodule::tata("arg1");
=head1 DESCRIPTION
La fonction toto prend en arguments deux chaînes de caractères
et retourne la concaténation de ces chaînes.
```

Ces commentaires peuvent ensuite être traités par divers programmes tels que pod2html, où encore **perldoc**, essayez : `perldoc Monmodule`.

Remarque importante

Dans le prochain cours, nous abordons la programmation orientée objet en perl. Il est bon de savoir qu'une classe contient la définition d'un type d'objets, et qu'en perl, une classe n'est autre qu'un module *spécial* (voir transparents).