Cycle de formation Meso@LR

Environment Module

Environment Module est un logiciel écrit en TCL qui permet à un utilisateur de configurer son environnement shell selon ses besoins. *Environment module* permet de faire cohabiter plusieurs configurations logicielles sur une même machine (par exemple plusieurs versions de gcc). Il est bien adapté aux environnements multi-utilisateurs tel que les clusters de calcul.

Environment Module est constitué d'une commande module et de fichiers de configuration modulefile.

Exemple :

```
$ python --version
$ module load python/3.7.2
$ python --version
```

La commande module va modifier les variables d'environnement en fonction de directives décrites dans les fichiers modulefiles.

Les deux principales variables d'environnement sont PATH (pour les exécutable) et LD_LIBRARY_PATH (pour les bibliothèques).

```
$ which python
$ echo $PATH
$ module rm python/3.7.2
$ which python
$ echo $PATH
```

Remarque : la lecture des PATH est séquentielle.

Attention : Les modifications apportées par l'utilisation de modules sont locales aux shell depuis lesquels elles ont été exécutées.

Format de la commande module (principales options) :

```
module [paramètre] <nom_du_module>
```

- av / avail : liste des modules disponibles
- display / show : affiche les informations d'un module
- add / load : charge un module

- list : liste des modules chargés
- rm / unload : décharge un module
- switch : remplace un module
- purge : décharge tous les modules
- help

Sur le cluster muse les modules sont installés dans une structure arborescente. Pour utiliser un module, il est nécessaire de pré-charger la branche dans laquelle il se trouve. Certaines branches sont pré-chargées automatiquement.

Environment Module dispose de ces propres variables d'environnement. Les utilisateurs ont accès aux modules (modul avail) via la variable d'environnement MODULEPATH

```
$ module purge
$ echo $MODULEPATH
$ module load cv-standard
$ echo $MODULEPATH
```

Personnalisation de Environment Module

La façon la plus simple de personnaliser son environnement est d'ajouter des commandes module dans les fichiers de configuration du shell (fichier .bash_profile pour le bash).

Le module use.own permet de créer son propre espace de module.

Au premier chargement, le module use.own crée un répertoire privatemodule ($HOME/privatemodules) dans lequel l'utilisateur peut créer ses propres modules ou adapter des modules existants.

L'exemple de fichier modulefile ci-dessous intègre un certain nombre de vérification (existence du répertoire, dépendance, …). Dans une version simplifiée seul l'entête et l'initialisation des variables PATH sont indispensables. Sans la ligne **`#%Module`** le fichier ne sera pas considéré comme un module par l'application.

```
#%Module
#
# @name:     monprog
# @version:  1.0
# @packaging: Baptiste Chapuisat
#

# Definie les variables internes au module
#-----------------------------------------
set   name      monprog
set   version   1.0
set   prefix    $::env(HOME)/F_HPC@LR
```

```
# S'affiche avec l'option help
#-----------------------------------------
proc ModulesHelp { } {
   puts stderr "\tExemple de documentation"
   puts stderr "\tpour le module [module-info name]"
   puts stderr ""
}

# Test le repertoire
# ----------------------------------------
if {![file exists $prefix]} {
   puts stderr "\t Load Error: $prefix does not exist"
   break
   exit 1
}

# Dependance
#
# prereq python/3.7.2
module load python/3.7.2

# Update common variables in the environment
# ----------------------------------------
prepend-path   PATH              $prefix/bin
prepend-path   LD_LIBRARY_PATH   $prefix/lib

setenv         MP_HOME           $version
```

Page du projet

http://modules.sourceforge.net/

Documentation officiel

https://modules.readthedocs.io/en/stable/index.html

Les variables d'environnement :

_LMFILES, MODULEPATH, LOADEDMODULES, MODULESHOME, MODULE_VERSION_STACK, MODULE_VERSION