Moduły środowiska

Z CIS

Artykuł ten opisuje docelową konfigurację systemu modułów. Obecnie środowisko modułów jest ładowane automatycznie (grupy module_utils, apps, tools oraz moduły cis-env, cis-auto-remove) i nie ma konieczności ładowania poszczególnych grup modułów ręcznie.

Moduły środowiska pozwalają w wygodny sposób zarządzać jakie oprogramowanie jest dostępne w aktywnej sesji użytkownika. Funkcjonalność którą oferują to:

- wyświetlenie dostępnych pakietów oprogramowania,
- aktywacja/deaktywacja wybranych pakietów,
- wylistowanie aktywnych pakietów,
- wyświetlanie szczegółowej informacji o pakietach.

Jeśli korzystamy z powłoki bash we wszystkich podanych niżej przykładach działa uzupełnianie składni z wykorzystaniem klawisza <Tab>. Dodatkowe informacje o składni/funkcjonalności polecenia module można uzyskać wywołując:

```
$ man module
```

UWAGA
Polecam zapoznać się z sekcją Znane problemy żeby nie tracić czasu na walkę z niedociągnięciami systemu moduł

Spis treści

Dostępne moduły

Dostępne pakiety oprogramowania użytkownik może wyświetlić wywołując komendę **module avail**.

```
$ module avail

------------------------------------------ /mnt/opt/modules/3.2.9/module_utils -----------------------------
cis-auto-remove dot             module-info     null
cis-env         module-cvs      modules         use.own

-------------------------------------------------- /mnt/opt/modules/3.2.9/tools ------------------------------
```

```
binutils/2.22                          gcc/4.6.3                              python/2.7.2-x86_64-gcc43(default)
blas/20070405a-x86_64-gcc43(default) lapack/3.1.1a-x86_64-gcc43(default)  python/2.7.2-x86_64-gcc46
blas/20110419-x86_64-gcc46           lapack/3.4.0-x86_64-gcc46            unison/2.32.52
cmake/2.8.6-x86_64-gcc43             nmon/14g(default)                    vim/7.3
flex/2.5.35                          nmon/14g-x86_64-gcc44                wine/1.5.4
gcc/4.3.2                            open64/4.5.1
gcc/4.4.6                            openmpi/1.4.3

------------------------------------------- /mnt/opt/modules/3.2.9/apps --------------------------------
fluent/14.0.1
--------------------------------------------------------------------------------------------------------
```

Możliwe jest zawężenie listy oprogramowania np:

```
$ module avail gcc

------------------------------------------- /mnt/opt/modules/3.2.9/tools --------------------------------
gcc/4.3.2 gcc/4.4.6 gcc/4.6.3
--------------------------------------------------------------------------------------------------------
```

Udostępnione pakiety zostały podzielone na trzy grupy:

- **module_utils** - pakiety narzędziowe ułatwiające pracę z systemem modułów
- **tools** - pakiety zawierające oprogramowanie narzędziowe: biblioteki, kompilatory, edytory, narzędzia systemowe etc.
- **apps** - pakiety z oprogramowaniem naukowym. Bardziej szczegółowe instrukcje wykorzystania poszczególnych pakietów naukowych znajdują się w Oprogramowanie naukowe

Format nazewnictwa pakietów jest następujący **<nazwa>/<wersja>-<architektura>**. Np.: python/2.7.2-x86_64-gcc46 to Python w wersji 2.7.2 skompilowany pod architekturę x86_64 z wykorzystaniem kompilatora GCC w wersji 4.6.*.

Dodatkowa informacja o pakietach

Dodatkową informację o pakietach możemy uzyskać poprzez komendy:

module whatis

```
$ module whatis

blas/20110419-x86_64-gcc46: BLAS (Basic Linear Algebra Subprograms) Library, Version: 20110419-x86_64-gcc46
cmake/2.8.6-x86_64-gcc43: CMake build system, Version: 2.8.6-x86_64-gcc43
flex/2.5.35          : Flex (The Fast Lexical Analyzer), Version: 2.5.35
gcc/4.3.2            : GNU Compiler Collection, Version: 4.3.2
```

```
$ module whatis python

python               : Python programming language, Version: 2.7.2-x86_64-gcc43
```

module help

```
$ module help python

---------- Module Specific Help for 'python/2.7.2-x86_64-gcc43' --------------------------

   Name: python
   Version 2.7.2
   Category: Compilers/Interpreter
   Platform: x86_64-gcc43
   URL http://python.org/
   Description: Python is a programming language that lets you work more
     quickly and integrate your systems more effectively. You can learn to use
     Python and see almost immediate gains in productivity and lower maintenance
```

```
     costs.
   Maintainer's email: konrad.klimaszewski@ncbj.gov.pl
```

module show

```
$ module show python

-------------------------------------------------------------------
/mnt/opt/modules/3.2.9/tools/python/2.7.2-x86_64-gcc43:

conflict         python
setenv           NCPUS 1
module-whatis    Python programming language, Version: 2.7.2-x86_64-gcc43
prereq   gcc/4.3.2
setenv           PYTHON_HOME /mnt/opt/tools/python/2.7.2-x86_64-gcc43
prepend-path     LD_LIBRARY_PATH /mnt/opt/tools/python/2.7.2-x86_64-gcc43/lib
prepend-path     PATH /mnt/opt/tools/python/2.7.2-x86_64-gcc43/bin
prepend-path     MANPATH /mnt/opt/tools/python/2.7.2-x86_64-gcc43/share/man
prepend-path     PYTHONPATH /mnt/opt/tools/python/2.7.2-x86_64-gcc43/lib/python2.7
-------------------------------------------------------------------
```

Listowanie aktualnie aktywnych pakietów

Aby sprawdzić jakie pakiety są aktualnie załadowane wykorzystujemy komendę **module list**

```
$ module list

Currently Loaded Modulefiles:
  1) cis-env                     3) gcc/4.3.2
  2) cis-auto-remove             4) python/2.7.2-x86_64-gcc43
```

Aktywacja modułów

Aby aktywować wybrany moduł wywołujemy komendę **module load**, np.: **module load <nazwa>** załaduje domyślą wersję pakietu <nazwa>. Aby załadować konkretną wersję należy wywołać **module load <nazwa>/<wersja>**

```
$ module load python

 'gcc/4.3.2' load complete.
 'python/2.7.2-x86_64-gcc43' load complete.
```

```
$ module load lapack/3.4.0-x86_64-gcc46

 'binutils/2.22' load complete.
 'gcc/4.6.3' load complete.
 'lapack/3.4.0-x86_64-gcc46' load complete.
```

Na powyższych przykładach widać że niektóre moduły posiadają zależności w postaci innych modułów. Jeśli jest to możliwe system spróbuje automatycznie załadować wszystkie zależności.

Deaktywacja modułów

Aby wyładować wybrany moduł wywołujemy komendę **module unload <nazwa>**.

```
$ module unload lapack

 'binutils/2.22' unload complete.
```

```
'gcc/4.6.3' unload complete.
'lapack/3.4.0-x86_64-gcc46' unload complete.
```

Automatyczna deaktywacja zależności które zostały załadowane przy aktywacji moduły następuje jedynie gdy aktywny jest moduł **cis-auto-remove**.

Automatyczna aktywacja modułów podczas logowania

Moduły mogą być aktywowane automatycznie podczas ładowania jeśli zdefiniujemy je w pliku *.bashrc* (lub odpowiednikowi dla innej powłoki). Przykładowy plik *.bashrc* aktywujący openmpi oraz fluent-a.

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# User specific aliases and functions

module load openmpi
module load fluent
```

Prywatne moduły

Możliwe jest tworzenie i wykorzystywanie własnych modułów (obsługujących prywatne wersje oprogramowania użytkownika). Pliki prywatnych modułów użytkownika należy umieścić w katalogu *$HOME/privatemodules*. Moduły te stają się widoczne dla systemu modułów po wywołaniu komendy:

```
$ module load use.own
```

Pliki modułów pisane są w języku Tcl, dokumentacja opisująca ich składnię dostępna jest na stronie http://modules.sourceforge.net oraz poprzez komendę

```
$ man modulefile
```

Moduły specjalne

Dwa z modułów dostępnych w grupie module_tools mają specjalne znaczenie:

- **cis-env** - definiuje zmienne środowiskowe niezbędne do prawidłowego działania większości modułów. Stąd moduł ten musi być zawsze aktywny. Należy pamiętać o aktywacji **cis-env** po wywołaniu komendy **module purge**
- **cis-auto-remove** - moduł ten steruje automatyczną dekatywacją modułów załadowanych jako zależności. Gdy jest aktywny automatyczna deaktywacja jest włączona, dekatywacja **cis-auto-remove** wyłącza tą funcjonalność.
- **cis-quiet** - załadowanie tego modułu wyłącza komunikaty generowane podczas aktywacji/deaktywacji modułów.

Znane problemy

- Funkcjonalność **module switch** jest w chwili obecnej popsuta. Jako rozwiązanie należy ręcznie

wywołać:

```
$ module unload <nazwa>
$ module load <nazwa>/<wersja>
```

- Po wywołaniu **module purge** należy wywołać **module load cis-env**
- Moduły OpenMPI oraz MVAPICH2 są poprawnie skonfigurowane **tylko** na węzłach klastra. Oznacza to, że na usrint nie ma możliwości uruchomienia poleceń **mpirun**, **mpiexec**, **mpicc** etc. Moduły te należy ładować z poziomu skryptu uruchamiającego zadanie lub będąc zalogowanym interaktywnie (qsub -I) na dowolnym węźle klastra.

Źródło „https://doc.cis.gov.pl/cis/index.php?title=Modu%C5%82y_%C5%9Brodowiska&oldid=464”
