Environment Modules

最終更新: 2025/1/27

はじめに

Environment Modules (module
コマンド)の利用は必須ではありませんが、手動で設定を行うよりも便利な場合があります。

注意点

- ログインシェルが bash 系で、ジョブスクリプトも sh, bash, zsh ならば特別な設定は必要ありません
- ログインシェルやジョブスクリプトが csh の場合は追加の設定が必要
    - csh のジョブスクリプトでは module コマンド利用前に source /etc/profile.d/modules.csh を実行する必要があります。
    - ログインシェルが csh の場合にはジョブスクリプトで /bin/sh を使う場合にも source /etc/profile.d/modules.sh (あるいは . /etc/profile.d/modules.sh)を実行する必要があります。

基本的な利用方法

moduleコマンドで操作します。フロントエンドサーバー、計算ノードで共通に利用できます。ただし、上記の通り csh
スクリプトでは特殊な注意が必要です。ログイン時に自動でいくつかのパッケージが読み込まれていることにもご注意くださ

module avail やアプリケーション一覧ページを参照して利用可能なパッケージを探し、module load
コマンドで必要なパッケージを読み込むことが基本となります。現在読み込んでいるパッケージは module list
で確認可能です。コマンドの使い方については次の項目もご確認ください。

アプリケーションによっては module
を用意されていない場合があります。そのようなアプリについては上記アプリケーション一覧ページや /apl
内を直接探してください。

コマンド

基本的なコマンドについては以下に簡単な例を掲載していますが、他にも様々なコマンドがあります。
公式ページの module subcommand のセクションを確認したり、ウェブ検索するなどしてご確認ください。

- load: 指定したモジュールを読み込む
- unload: 指定したモジュールを外す
- purge: 全モジュールのアンロード
- avail: 利用可能なモジュールの表示
- list: 現在読み込んでいるモジュールの表示
- whatis: モジュールの簡単な説明の表示
- save: 現在の設定の保存
- saverm: 保存した設定の削除

- -s オプションでメッセージを非表示にする
- module のデフォルトバージョン
- ジョブスクリプトでの利用例
    - /bin/sh スクリプトの場合
    - /bin/csh スクリプトの場合

load: 指定したモジュールを読み込む

指定したモジュールを読み込みます。バージョン名を省略することも可能です。その場合は指定されたデフォルトのバージ

実行例1: openmpi 4.1.6 (gcc12 でビルドしたもの)の読み込み

```
$ which mpirun
/usr/bin/which: no mpirun in (...)
$ module load openmpi/4.1.6/gcc12
Loading openmpi/4.1.6/gcc12
```

unload: 指定したモジュールを外す

モジュールのアンロードを行います(remove でも可)。依存関係が複雑なものについては unload を実行時に環境が破綻する可能性があります。そのような時は申し訳ありませんが、後述の purge コマンドで一旦初期化するか、一度ログアウトするなどしてから必要なモジュールを load し直してください。

purge

現在読み込んでいるモジュールをすべて unload します。

```
$ module list
Currently Loaded Modulefiles:
1) 2024        5) cuda/11.2        9) orca/5.0.4
```

avail: 利用可能なモジュールの表示

実

利用可能なモジュールの一覧を表示します。文字列を指定すれば、指定した文字列にマッチするモジュールだけを抽出しま

```
$ module avail
... apl/modulefiles/defaults ...
2022 2024
```

list: 現在読み込んでいるモジュールの表示

実

既に上の方の実行例でも頻繁に使用していますが、現在読み込んでいるモジュールのリストを表示できます。

```
$ module list
Currently Loaded Modulefiles:
1) gcc/4.9.0  2) cuda/12.0   3) 2022
```

whatis: モジュールの簡単な説明の表示

モジュールの簡単な説明を表示します。モジュール名を指定しない場合は全モジュールの情報を表示します。

```
$ module whatis mkl/2022
------- /apl/modulefiles/apl -------
   mkl/2022.0.2: Intel(R) oneAPI Math Kernel Library (oneMKL) IA-64 architecture
   mkl/2022.2.1: Intel(R) oneAPI Math Kernel Library (oneMKL) IA-64 architecture
$ module whatis
(??)
```

save: 現在の設定の保存

現在の設定を保存することができます。名前を省略すると default が使われます。default の内容は次回ログインした時に自動で読み込まれます(module restore default)。自身で作成した module を読み込んだ状態で save することもできます。保存した内容は ~/.module に保存され、module restore (名前) で読み込めます。

実

(キ

```
$ module list
Currently Loaded Modulefiles:
1) tbb/2021.7.1          4) cuda/11.2              7) amber/20u13
2) compiler-rt/2022.2.1   5) pbs/22.05.11
3) mkl/2022.2.1          6) openmpi/4.1.4-hpcx/gcc9

Key:
default-version  auto-loaded
$ module save
$ module purge
$ module list
No Modulefiles Currently Loaded.
$ module restore
Loading tbb/2021.7.1
```

```
Loading compiler-rt/2022.2.1
Loading mkl/2022.2.1
Loading cuda/11.2
Loading pbs/22.05.11
Loading openmpi/4.1.4-hpcx/gcc9
Loading amber/20u13
```

saverm: 保存した設定の削除

保存した設定を削除します。名前を省略すると default が対象になります。

-s オプションでメッセージを非表示にする

```
$ module saverm testsave
$ module saverm
```

依存関係を満たすために追加の module が読み込まれた場合などには追加のメッセージが出力されますが、module コマンドに -s オプションを追加することで出力を抑制できます。以下は load の例ですが、unload や purge 等他のオプションでも有効です。

-s オプションを追加した場合はエラーメッセージも抑制される点にご注意ください

```
$ module purge
$ module avail openmpi
------------------------ /apl/modules/util ------------------------
openmpi/3.1.6/aocc3              openmpi/4.1.5-hpcx2.16/icc2023.1.0
openmpi/3.1.6/aocc4              openmpi/4.1.5-hpcx2.16/icx2023.1.0
openmpi/3.1.6/gcc8               openmpi/4.1.5/aocc4
...
openmpi/4.1.4-hpcx/gcc10         openmpi/4.1.6/aocc4.1
openmpi/4.1.4-hpcx/gcc11         openmpi/4.1.6/gcc8
openmpi/4.1.4-hpcx/gcc12         openmpi/4.1.6/gcc11
...

Key:
modulepath  directory/  default-version
$ module load openmpi
Loading openmpi/4.1.6/gcc8
  Loading requirement: pbs/22.05.11
$ module list
Currently Loaded Modulefiles:
 1) pbs/22.05.11   2) openmpi/4.1.6/gcc8(default:openmpi)

Key:
(symbolic-version)  auto-loaded  default-version
```

moduleをunloadする際にも簡略化した表記が有効です。

```
$ module list
Currently Loaded Modulefiles:
 1) pbs/22.05.11   2) openmpi/4.1.6/gcc8(default:openmpi)

Key:
(symbolic-version)  auto-loaded  default-version
$ module unload openmpi
```

```
Unloading openmpi/4.1.6/gcc8
  Unloading useless requirement: pbs/22.05.11
$ module list
```

ジョブスクリプトでの利用例

/bin/sh スクリプトの場合

```
#!/bin/sh
#PBS select=ncpus=1:mpiprocs=1:ompthreads=1:ngpus=1
#PBS -l walltime=72:00:00

if [ ! -z $PBS_O_WORKDIR ]; then
  cd $PBS_O_WORKDIR
fi

module -s purge
module -s load amber/20u13

pmemd.cuda -O -i mdin ......
```

ログインシェルが csh で、ジョブスクリプトだけsh, bashの場合は、module -s purge コマンドより前の行に source /etc/profile.d/modules.sh が必要です。purge コマンドで一度全ての module を解除した後で、実際に必要な module を読み込みます。これが基本形です。なお、ここでは module -s で出力を消しているため、もし読み込みに失敗していてもエラーが出ません。module の読み込みに失敗している可能性がある場合は -s を消して再実行し、出力内容を確認した方が良いでしょう。

/bin/csh スクリプトの場合

cshの場合、moduleコマンドは /etc/profile.d/modules.csh でaliasとして定義されています。
ログインして対話型シェルで使う場合には特に設定は要りませんが、
ジョブスクリプトで利用する場合には明示的に /etc/profile.d/modules.csh を読み込む必要があります。
それ以外の部分については bash の場合と同様です。

```
#!/bin/csh -f
#PBS select=ncpus=1:mpiprocs=1:ompthreads=1:ngpus=1
#PBS -l walltime=72:00:00

if ( $?PBS_O_WORKDIR ) then
  cd $PBS_O_WORKDIR
endif

source /etc/profile.d/modules.csh # required!
module -s purge
module -s load amber/20u13

pmemd.cuda -O -i mdin .......
```

参考リンク

- サンプルジョブの実行
- アプリケーションライブラリの構築方法
- Environment Modules の公式サイト(英語)