

Olaf Cluster User Training

환경 모듈(environment module)



ESNET[®]

심희찬

11, Nov 2024

목차

- ▶ **환경 모듈(Environment Module)이란?**

- ▶ 환경 모듈(Environment Module)의 필요성
- ▶ 환경 모듈(Environment Module)의 종류

- ▶ **환경모듈 사용법**

- ▶ **환경 모듈 구성 방법**

- ▶ **실습**

환경 모듈이란?

환경 모듈(Environment Module)이란 ?

- 환경 모듈(Environment Modules)은 컴퓨팅 환경에서 사용자가 다양한 소프트웨어 패키지를 손쉽게 관리하고 불러올 수 있도록 돕는 시스템입니다. 일반적으로 고성능 컴퓨팅(HPC) 환경이나 연구용 서버에서 주로 사용되며, 사용자는 환경 모듈을 통해 필요한 소프트웨어와 관련 **환경 변수**를 동적으로 로드하고 설정할 수 있습니다.

환경 모듈(Environment Module)의 필요성 ?

1. 버전 관리
2. 경로 설정의 간소화
3. 일관된 환경 제공

환경 모듈(Environment Module)의 필요성 ?

1. 버전 관리: 환경 모듈을 사용하면 여러 버전의 소프트웨어를 동시에 설치하고 쉽게 전환할 수 있습니다. 사용자는 module load 명령어로 필요한 버전의 소프트웨어를 선택하여 사용할 수 있어, 다른 사용자의 환경에 영향을 주지 않고 독립적으로 소프트웨어를 이용할 수 있습니다.

예시) intel compiler 버전 관리

```
$ module ava intel
```

```
----- /opt/ibs_lib/modulefiles/compiler -----
```

```
intel/2021.2.0 intel/2021.3.0(default) intel/2022.0.2 intel/2022.2.1
```

환경 모듈(Environment Module)의 필요성 ?

2. 경로 설정의 간소화: 환경 모듈은 PATH, LD_LIBRARY_PATH, CPATH, MANPATH와 같은 환경 변수를 자동으로 설정해주어 사용자가 일일이 경로를 설정할 필요가 없습니다. 따라서 소프트웨어를 설치하거나 실행할 때 발생할 수 있는 경로 관련 오류를 줄일 수 있습니다.

예시) 'mpirun' 명령어를 실행시켜야 할 때

모듈 적재 전

```
$ /opt/ibs_lib/apps/rocky/openmpi/4.1.1/INTEL/202130/bin/mpirun [command]
```

모듈 적재 후

```
$ module load intel/2021.3.0 openmpi/4.1.1-Rocky
```

```
$ mpirun [command]
```

환경 모듈(Environment Module)의 필요성 ?

3. **일관된 환경 제공:** 모든 사용자에게 동일한 환경을 제공함으로써, 호환성 문제나 환경 설정에 따른 에러 발생을 최소화할 수 있습니다. 연구나 프로젝트 팀이 같은 소프트웨어 환경에서 작업할 수 있어 공동 작업이 수월해집니다.

예시) 일관된 모듈 사용

```
edu01 $ module load intel/2021.3.0 impi/2021.7.1
```

```
edu01 $ module list
```

Currently Loaded Modulefiles:

1) intel/2021.3.0(default) 2) impi/2021.7.1(default)

```
edu02 $ module load intel/2021.3.0 impi/2021.7.1
```

```
edu02 $ module list
```

Currently Loaded Modulefiles:

1) intel/2021.3.0(default) 2) impi/2021.7.1(default)

환경 모듈의 종류

- LMOD (Lua-based 모듈)
- Environment Module (환경 모듈)

특성	LMOD	Environment Module
언어 기반	Lua	Tcl (Tool Command Language)
지원하는 언어	Lua, Tcl	Tcl
자동 로드 기능	모듈 간의 의존성 관리가 가능하며, 로드할 때 의존하는 모듈을 자동으로 로드함	기본적으로는 수동으로 의존 모듈을 로드해야 함
모듈 계층화	소프트웨어 스택을 계층화하여 상위 모듈이 특정 하위 모듈에 의존할 때 자동으로 계층적 로드 가능	일반적으로 계층화 기능이 없음
의존성 관리	의존성과 충돌을 자동으로 관리할 수 있어, 중복 또는 충돌하는 모듈을 효과적으로 방지	충돌 또는 중복 관리 기능이 제한적
캐싱 기능	모듈 캐싱 기능을 통해 모듈 탐색 속도 향상	별도의 캐싱 기능이 없음

환경 모듈 사용법

환경 모듈 사용법

1. module [ava | avail]
 - MODULEPATH에 등록된 모듈들을 보여주는 명령어
2. module list
 - 현재 Load된 모듈의 목록
3. module [add | load] Modulefile
 - 셸 환경에 모듈 파일을 로드
4. module [unload | rm | remove] Modulefile
 - 셸 환경에서 모듈 파일을 제거
5. module [switch | swap] Modulefile1 Modulefile2
 - 로드된 modulefile1을 modulefile2로 전환
6. module [display | show] Modulefile
 - 모듈 파일에 대한 정보를 표시
7. module purge
 - 로드된 모든 모듈 파일을 언로드
8. module use [-a | --append] directory
 - MODULEPATH 환경 변수에 하나 이상의 디렉토리를 추가
9. module help Modulefile
 - 모듈 파일에 대한 도움말 조회

환경 모듈 사용법 – “module avail”

```
$ module [ ava | avail ]
```

- MODULE_PATH에 등록된 모듈들을 보여주는 명령어

```
edu@olaf1 $ module ava
----- /opt/ibs_lib/modulefiles/compilers -----
gcc/8.5.0  gcc/11.2.0  intel/2021.2.0      intel/2022.0.2  pgi/23.5
gcc/9.3.0  gcc/12.2.0  intel/2021.3.0(default)  intel/2022.2.1

----- /opt/ibs_lib/modulefiles/mpi -----
impi/2021.1.1  impi/2021.3.0  impi/2021.7.1(default)  openmpi/4.1.1-Rocky
impi/2021.2.0  impi/2021.5.1  openmpi/4.1.1           openmpi/4.1.4

----- /opt/ibs_lib/modulefiles/libraries -----
anaconda/23.09.0  cudatoolkit/12.2  ghostscript/10.02.1  lapack/3.11.0      openssl/1.1.1g  readline/8.2
bazel/1.2.2      cudnn/8.9.7.29-cuda12  git-lfs/3.5.1      libtirpc/1.3.3      pcre2/10.42     root/6.26.10
bison/3.8.2      curl/7.61.1(default)  gmp/6.2.1          libxml2/2.11.4      petsc/3.18.2    singularity/4.1.1
blas/3.11.0      curl/7.88.1         go/1.22.0          lustre/2.15.1       pnetcdf/1.12.3  sqlite/3.43.0
boost/1.65.1     difx/2.6.3         gromacs/2024.3     miniconda/23.01.0   proj/8.2.1      trilinos/13.4.1
boost/1.81.0     difx/2.8.1         gsl/2.7.1          mpc/1.3.1           python/3.6.10   ucx/1.13.1
bzip2/1.0.8      ECW/5.4.0          hdf4/4.2.14        mpfr/4.1.1          python/3.7.2    wgrib/1.8.2
charm/7.0.0      fftw/3.3.8         hdf4/4.2.15        ncurses/6.4         python/3.8.16   wgrib2/3.1.1
cmake/3.18.4     fftw/3.3.10        hdf5/1.12.1        ncview/2.1.10       python/3.9.16   xz/5.4.2
cmake/3.28.1     flex/2.6.4         isl/0.24           netcdf-fortran/4.5.4  qchem/6.0.1     zlib/1.2.11(default)
cudatoolkit/11.7  gaussian/g16.c01    jasper/3.0.6       netcdf/4.8.1        qmcpack/3.16.0  R/4.0.5
cudatoolkit/11.8  gdal/3.5.0         jpeg/9e            openfoam/v2006

----- /opt/ibs_lib/modulefiles/cryoem -----
relion-gpu/4.0.0  relion/4.0.0

----- /opt/ibs_lib/modulefiles/protodsign -----
DL-Binder-Design/1.0.0  protein-miniconda/23.1.0
```

환경 모듈 사용법 – “module avail”

\$ module [ava | avail] [Modulefile]

```
edu@olaf1 $ module avail intel impi fftw
----- /opt/ibs_lib/modulefiles/compilers -----
intel/2021.2.0  intel/2021.3.0(default)  intel/2022.0.2  intel/2022.2.1

----- /opt/ibs_lib/modulefiles/libraries -----
fftw/3.3.8  fftw/3.3.10
edu@olaf1 $ module avail i
----- /opt/ibs_lib/modulefiles/compilers -----
intel/2021.2.0  intel/2021.3.0(default)  intel/2022.0.2  intel/2022.2.1

----- /opt/ibs_lib/modulefiles/mpi -----
impi/2021.1.1  impi/2021.2.0  impi/2021.3.0  impi/2021.5.1  impi/2021.7.1(default)

----- /opt/ibs_lib/modulefiles/libraries -----
isl/0.24
```

환경 모듈 사용법 – “module list”

\$ module list

- 현재 Load된 모듈의 목록

```
edu@olaf1 $ module list
Currently Loaded Modulefiles:
 1) intel/2022.0.2    4) gmp/6.2.1          7) cudatoolkit/12.2  10) libxml2/2.11.4
 2) impi/2021.5.1    5) git-lfs/3.5.1       8) libtirpc/1.3.3    11) jpeg/9e
 3) go/1.22.0        6) ghostscript/10.02.1 9) jasper/3.0.6
```

환경 모듈 사용법 – “module load”

\$ module [add | load] Modulefile

- 셸 환경에 모듈 파일을 로드

```
edu@olaf1 $ module ava intel impi
----- /opt/ibs_lib/modulefiles/compilers -----
intel/2021.2.0  intel/2021.3.0(default)  intel/2022.0.2  intel/2022.2.1

----- /opt/ibs_lib/modulefiles/mpi -----
impi/2021.1.1  impi/2021.2.0  impi/2021.3.0  impi/2021.5.1  impi/2021.7.1(default)

edu@olaf1 $ module load intel/2022.0.2 impi/2021.5.1

edu@olaf1 $ module list
Currently Loaded Modulefiles:
 1) intel/2022.0.2  2) impi/2021.5.1
```

환경 모듈 사용법 – “module unload”

\$ module [unload | rm | remove] Modulefile

- 셸 환경에서 모듈 파일을 제거

```
edu@olaf1 $ module list
Currently Loaded Modulefiles:
  1) intel/2022.0.2    2) impi/2021.5.1

edu@olaf1 $ module unload impi/2021.5.1

edu@olaf1 $ module list
Currently Loaded Modulefiles:
  1) intel/2022.0.2
```

환경 모듈 사용법 – “module switch”

\$ module [switch | swap] Modulefile1 Modulefile2

- 로드된 modulefile1을 modulefile2로 전환

```
edu@olaf1 $ module list
Currently Loaded Modulefiles:
 1) intel/2022.0.2    2) impi/2021.7.1(default)

edu@olaf1 $ module switch impi/2021.7.1 impi/2021.2.0

edu@olaf1 $ module list
Currently Loaded Modulefiles:
 1) intel/2022.0.2    2) impi/2021.2.0
```

환경 모듈 사용법 – “module show”

\$ module [display | show] Modulefile

- 모듈 파일에 대한 정보를 표시

```
edu@olaf1 $ module show impi/2021.2.0
-----
/opt/ibs_lib/modulefiles/mpi/impi/2021.2.0:

conflict      impi
conflict      openmpi
conflict      mpich
conflict      mvapich2
module-whatis  {Intel(R) MPI Library}
setenv        I_MPI_ROOT /opt/ibs_lib/apps/oneapi/mpi/2021.2.0
setenv        MPI_ROOT /opt/ibs_lib/apps/oneapi/mpi/2021.2.0
prepend-path  CLASSPATH /opt/ibs_lib/apps/oneapi/mpi/2021.2.0/lib/mpi.jar
prepend-path  PATH /opt/ibs_lib/apps/oneapi/mpi/2021.2.0/bin
prepend-path  LD_LIBRARY_PATH /opt/ibs_lib/apps/oneapi/mpi/2021.2.0/lib/release:/opt/ibs_lib/apps/oneapi/mpi/2021.2.0/
lib
prepend-path  LIBRARY_PATH /opt/ibs_lib/apps/oneapi/mpi/2021.2.0/lib/release:/opt/ibs_lib/apps/oneapi/mpi/2021.2.0/lib
prepend-path  CPATH /opt/ibs_lib/apps/oneapi/mpi/2021.2.0/include
prepend-path  MANPATH /opt/ibs_lib/apps/oneapi/mpi/2021.2.0/man
setenv        UCX_IB_SL 3
setenv        I_MPI_PMI_LIBRARY /usr/lib64/libpmi2.so
setenv        I_MPI_PIN_RESPECT_CPUSET off
setenv        I_MPI_HYDRA_BOOTSTRAP slurm
setenv        MPI_ENV IMPI
setenv        MPI_VER 202120
```

환경 모듈 사용법 – “module purge”

\$ module purge

- 로드된 모든 모듈 파일을 언로드

```
edu@olaf1 $ module list  
Currently Loaded Modulefiles:  
  1) intel/2022.0.2   2) impi/2021.2.0
```

```
edu@olaf1 $ module purge
```

```
edu@olaf1 $ module list  
No Modulefiles Currently Loaded.
```

환경 모듈 사용법 – “module use”

\$ module use [-a | --append] Module_directory

- MODULE_PATH 환경 변수에 하나 이상의 디렉토리를 추가

```
edu@olaf1 $ module use -a /opt/ibs_lib/apps/eb/modules/compiler
```

```
edu@olaf1 $ module ava
```

```
----- /opt/ibs_lib/apps/eb/modules/compiler -----  
GCC/12.3.0      GCCcore/12.3.0      intel-compilers/2023.1.0
```

환경 모듈 사용법 – “module help”

\$ module help Modulefile

- 모듈 파일에 대한 도움말 조회

```
edu@olaf1 $ module help intel
```

```
-----  
Module Specific Help for /opt/ibs_lib/modulefiles/compilers/intel/2021.3.0:
```

```
    This module is for use of Intel Compiler 2018.
```

```
    It needs module(s):
```

```
                None
```

```
    Use example:
```

```
        $ module load intel/2021.3.0
```

```
    Additional info:
```

```
        1. We can use Intel Advisor, Vtune, Inspector, TBB and MKL.
```

```
        2. Major environment variables is set up like these:
```

```
            CC=icc   CXX=icpc   FC=ifort   F77=ifort   F90=ifort
```

```
-----
```

환경 모듈 구성 방법

환경 모듈 구성 방법 - 환경 변수

MODULEPATH

환경 모듈 시스템이 모듈 파일을 찾는 경로를 정의하는 변수.

PATH

시스템이 명령어나 실행 파일의 경로를 정의하는 변수.

환경 모듈에서는 특정 프로그램의 바이너리 파일이 있는 디렉토리를 추가하거나 제거할 때 사용됩니다.

LIBRARY_PATH

컴파일할 때 정적 라이브러리(.a 파일) 위치를 정의하는 변수.

컴파일러가 정적 라이브러리를 찾을 수 있도록 경로를 설정하는 데 사용됩니다.

LD_LIBRARY_PATH

프로그램이 동적 라이브러리(.so 파일)를 찾는 경로를 정의하는 변수.

필요한 라이브러리가 설치된 경로를 지정하여 프로그램 실행 시 찾을 수 있도록 합니다.

CPATH

컴파일러가 헤더 파일을 찾는 경로를 정의하는 변수.

MANPATH

man 명령어가 매뉴얼 페이지를 찾는 경로를 정의하는 변수.

PKG_CONFIG_PATH

pkg-config 명령어가 라이브러리 구성 파일(.pc 파일)을 찾는 경로를 정의하는 변수.

환경 모듈 구성 방법 - 환경 변수

환경 변수 조회 명령어 : export, env

1. export

\$ export | grep [변수명]

```
edu@olaf1 $ export | grep MODULEPATH
declare -x MODULEPATH="/etc/scl/modulefiles:/opt/ibs_lib/modulefiles/compilers:/opt/ibs_lib/modulefiles/mpi:/opt/ibs_lib/modulefiles/libraries:/opt/ibs_lib/modulefiles/cryoem:/opt/ibs_lib/modulefiles/protdesign:/usr/share/Modules/modulefiles:/etc/modulefiles:/usr/share/modulefiles:/opt/cryoem/group/bmttest"
declare -x MODULEPATH_modshare="/opt/ibs_lib/modulefiles/libraries:1:/opt/cryoem/group/bmttest:1:/opt/ibs_lib/modulefiles/compilers:1:/opt/ibs_lib/modulefiles/mpi:1:/opt/ibs_lib/modulefiles/cryoem:1:/opt/ibs_lib/modulefiles/protdesign:1:/usr/share/Modules/modulefiles:1:/etc/modulefiles:1:/usr/share/modulefiles:1"
```

2. env

\$ env | grep [변수명]

```
edu@olaf1 $ env | grep -w PATH
PATH=/proj/home/ibs/bmttest/bmttest/.local/bin:/proj/home/ibs/bmttest/bmttest/bin:/usr/share/Modules/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin
```

환경 모듈 구성 방법 - 파일 위치

1. 모듈 파일의 위치

- Olaf 기준 모듈 파일의 위치는 "/opt/ibs_lib/modulefiles" 밑에 위치합니다
- 특정 경로에 모듈 파일을 생성하여 위치할 수 있습니다.

```
edu@olaf1 $ pwd
/opt/ibs_lib/modulefiles

edu@olaf1 $ ls
compilers  cryoem  example  gpulibraries  libraries  mpi  protdesign  python

edu@olaf1 $ ls libraries/
alphafold2  charm      ECW      gmp      jasper    mpc      openssl  qmcpack   ucx
anaconda   cmake      fftw     go       jpeg      mpfr     pcre2    R         wgrib
bagel      cudatoolkit flex      gromacs  lapack    ncurses  petsc    readline  wgrib2
bison      cudnn     gaussian gsl       libtirpc  ncview   pnetcdf  root      xz
blas       curl      gdal     hdf4     libxml2   netcdf   proj     singularity  zlib
boost     difx      ghostscript hdf5     lustre    netcdf-fortran  python  sqlite
bzip2     easybuild git-lfs  isl      miniconda openfoam  qchem    trilinos
```

환경 모듈 구성 방법 - 기본 구조

2. 모듈 파일의 기본 구조

- 환경 모듈 파일은 기본적으로 소프트웨어 경로와 환경 변수들을 설정합니다. ([Modules Specific Tcl Commands](#))

The diagram illustrates the basic structure of a Tcl module file. It features a black background with yellow and white text representing the code. Red boxes highlight specific parts of the code, and red arrows point from Korean annotations on the left to these highlighted sections.

```
#%Module
proc ModulesHelp { } {
    puts stderr "This module loads Example Software 1.0"
}
module-whatis "Example Software 1.0"

set EXAMPLE_HOME /path/to/example/1.0
setenv GL_EXAMPLE_HOME ${EXAMPL_HOME}
prepend-path PATH ${EXAMPLE_HOME}/bin
```

Annotations on the left side:

- Tcl 스크립트를 인식시키기 위한 명령어 (points to `#%Module`)
- module help 명령어에서 출력되는 내용 (points to `proc ModulesHelp { } {`)
- module show 명령어에서 출력되는 내용 (points to `module-whatis "Example Software 1.0"`)
- 모듈 파일내에서 사용되는 변수 지정 (points to `set EXAMPLE_HOME`)
- 셸 환경 변수 지정 (points to `setenv GL_EXAMPLE_HOME`)
- 지정한 경로를 기존 경로 앞에 추가하는 명령어 (points to `prepend-path PATH`)

환경 모듈 구성 방법

set 과 setenv 명령어의 차이

명령어	설명	유효 범위	사용 목적
set	일반 Tcl 변수 설정 예제	모듈 파일 내부	모듈 내에서만 사용되는 임시 변수 설정
setenv	<pre># 일반 변수 설정 (모듈 내부 전용) set software_home "/path/to/software" prepend-path PATH \${software_home}/bin # 환경 변수 설정 (사용자 세션에 적용됨) setenv SOFTWARE_HOME \$software_home prepend-path PATH \$env(SOFTWARE_HOME)/bin</pre>		환경 변수

환경 모듈 구성 방법 - 기본 구조

3. 프로그램의 기본 구조

```
edu@olaf1 $ tree -L 1
```

```
.
├── bin
├── include
├── lib
└── share
```

<일반적인 프로그램의 구조>

- modulefile 파일에서 프로그램의 파일 구조에 따라 수정을 해야만 합니다.

예시)

```
prepend-path CPATH=[PROGRAM_PATH]/include:$CPATH
```

```
prepend-path LIBRARY_PATH=[PROGRAM_PATH]/lib:$LIBRARY_PATH
```

```
prepend-path LD_LIBRARY_PATH=[PROGRAM_PATH]/lib:$LD_LIBRARY_PATH
```

```
prepend-path PATH=[PROGRAM_PATH]/bin:$PATH
```

```
prepend-path MANPATH=[PROGRAM_PATH]/man:$MANPATH
```

환경 모듈 구성 방법

```
edu@olaf1 $ tree /opt/ibs_lib/apps/example/2024.11.11/  
/opt/ibs_lib/apps/example/2024.11.11/  
├── bin  
│   ├── test.txt  
│   ├── tool1  
│   └── tool2  
├── include  
│   └── add.h  
├── lib  
│   └── libadd.so  
├── man  
│   └── man1  
│       ├── tool1.1  
│       └── tool2.1  
└── src  
    ├── libadd.c  
    ├── tool1.c  
    └── tool2.c
```

프로그램 경로:

/opt/ibs_lib/apps/example/2024.11.11

tool1 : 5와 10을 더하는 프로그램

tool2 : text 파일을 출력해주는 프로그램

환경 모듈 구성 방법

```
edu@olaf1 $ tree /opt/ibs_lib/apps/example/2024.11.11/
/opt/ibs_lib/apps/example/2024.11.11/
├── bin
│   ├── test.txt
│   ├── tool1
│   └── tool2
├── include
│   └── add.h
├── lib
│   └── libadd.so
├── man
│   └── man1
│       ├── tool1.1
│       └── tool2.1
└── src
    ├── libadd.c
    ├── tool1.c
    └── tool2.c
```

모듈 설정 없이 tool1 실행 절차

1. CPATH 변수 지정
export CPATH=/opt/ibs_lib/apps/example/2024.11.11/include:\$CPATH
2. LD_LIBRARY 변수 지정
export LD_LIBRARY_PATH=/opt/ibs_lib/apps/example/2024.11.11/lib:\$LD_LIBRARY_PATH
3. PATH 변수 지정
export PATH=/opt/ibs_lib/apps/example/2024.11.11/bin:\$PATH
4. MANPATH 지정
export MANPATH=/opt/ibs_lib/apps/example/2024.11.11/man:\$MANPATH
5. 실행

```
edu@olaf1 $ tool1
The sum of 5 and 10 is: 15
```

환경 모듈 구성 방법

```
edu@laf1:~$ tree /opt/ibs_lib/apps/example/2024.11.11/
/opt/ibs_lib/apps/example/2024.11.11/
├── include
├── lib
├── bin
├── man
├── src
└── tool2.c
```

소스 빌드 중에 함수를 찾지 못하여 빌드 진행이 안될 수 있음

동적 라이브러리를 찾지 못하여 프로그램이 실행이 안될 수 있음

파일 위치를 직접 찾아서 절대경로를 넣어줘야함

실행파일에 대한 설명 조회 불가능

모듈 파일 없이 tool1 실행 절차

1. CPATH 변수 지정

`export CPATH=/opt/ibs_lib/apps/example/2024.11.11/include:$CPATH`

2. LD_LIBRARY 변수 지정

`export LD_LIBRARY_PATH=/opt/ibs_lib/apps/example/2024.11.11/lib:$LD_LIBRARY_PATH`

3. PATH 변수 지정

`export PATH=/opt/ibs_lib/apps/example/2024.11.11/bin:$PATH`

4. MANPATH 변수 지정

`export MANPATH=/opt/ibs_lib/apps/example/2024.11.11/man:$MANPATH`

5. 실행

```
edu@laf1 $ tool1  
The sum of 5 and 10 is: 15
```

환경 모듈 구성 방법

```
edu@olaf1 $ tree /opt/ibs_lib/apps/example/2024.11.11/  
/opt/ibs_lib/apps/example/2024.11.11/  
├── bin  
│   ├── test.txt  
│   ├── tool1  
│   └── tool2  
├── include  
│   └── add.h  
├── lib  
│   └── libadd.so  
├── man  
│   └── man1  
│       ├── tool1.1  
│       └── tool2.1  
└── src  
    ├── libadd.c  
    ├── tool1.c  
    └── tool2.c
```

모듈 파일을 이용하여 tool1 실행 절차

1. Module PATH 추가
\$ module use -a /opt/ibs_lib/apps/example
2. 모듈 파일 적재
\$ module load 2024.11.11
3. 실행

```
edu@olaf1 $ tool1  
The sum of 5 and 10 is: 15
```

```
edu@olaf1 $ tool2 test.txt  
This module loads Example 1.0 environment
```

실습

실습

1. 모듈 파일 경로 지정
\$ mkdir -p ~/modulefiles/example
2. 모듈 파일 생성 및 작성
\$ vi ~/modulefiles/example/2024.11.11
3. MODULEPATH 추가 (module use)
\$ module use -a ~/modulefiles
4. module 조회 (module ava)
\$ module ava exam
5. module 적재 (module load)
\$ module load example/2024.11.11
6. 실행
\$ module help example/2024.11.11
\$ module show example/2024.11.11
\$ env | grep -w LD_LIBRARY_PATH
\$ tool1
\$ tool2 [text.file]

예시 파일 경로 : /opt/ibs_lib/modulefiles/example/2024.11.11

```
##Module
proc ModulesHelp { } {
    puts stderr "This module loads Example 1.0 environment"
}

module-whatis "Example 1.0 environment module"

# 경로 설정
set software_home "/opt/ibs_lib/apps/example/2024.11.11"

if {![file exists $software_home]} {
    puts stderr "Load Error: $software_home does not exist"
    break
    exit 1
}

# 실행 파일 경로
prepend-path PATH $software_home/bin

# 라이브러리 경로
prepend-path LIBRARY_PATH $software_home/lib
prepend-path LD_LIBRARY_PATH $software_home/lib

# 헤더 파일 경로
prepend-path CPATH $software_home/include

# 매뉴얼 경로
prepend-path MANPATH $software_home/man
```

참고사항

/opt/ibs_lib/modulefiles

```
edu@olaf1 $ pwd
/opt/ibs_lib/modulefiles

edu@olaf1 $ ls
compilers  cryoem  example  gpulibraries  libraries  mpi  protdesign  python

edu@olaf1 $ ls libraries/
alphafold2  charm      ECW      gmp      jasper    mpc      openssl  qmcpack    ucx
anaconda   cmake      fftw     go       jpeg      mpfr     pcre2    R           wgrib
bagel      cudatoolkit flex      gromacs  lapack    ncurses  petsc    readline   wgrib2
bison      cudnn      gaussian gsl       libtirpc  ncview   pnetcdf  root       xz
blas       curl       gdal     hdf4     libxml2   netcdf   proj      singularity zlib
boost      difx       ghostscript hdf5     lustre    netcdf-fortran python     sqlite
bzip2      easybuild  git-lfs  isl      miniconda openfoam  qchem     trilinos
```

<예시 모듈파일>

```
##Module1.0

proc ModulesHelp { } {
    puts stderr " "
    puts stderr "\tThis module is for use of [module-info name]."
    puts stderr " "
    puts stderr "\tuse example:"
    puts stderr "\t    $ module load [module-info name]"
    puts stderr " "
}

module load gcc/12.2.0 openmpi/4.1.1 gsl/2.7.1 bison/3.8.2 libtirpc/1.3.3 fftw/3.3.8

conflict intel

set name      difx
set version  2.6.3
set prefix    /opt/ibs_lib/apps/difx/2.6.3

if {[file exists $prefix]} {
    puts stderr "\t[module-info name] Load Error: $prefix does not exist"
    break
    exit 1
}

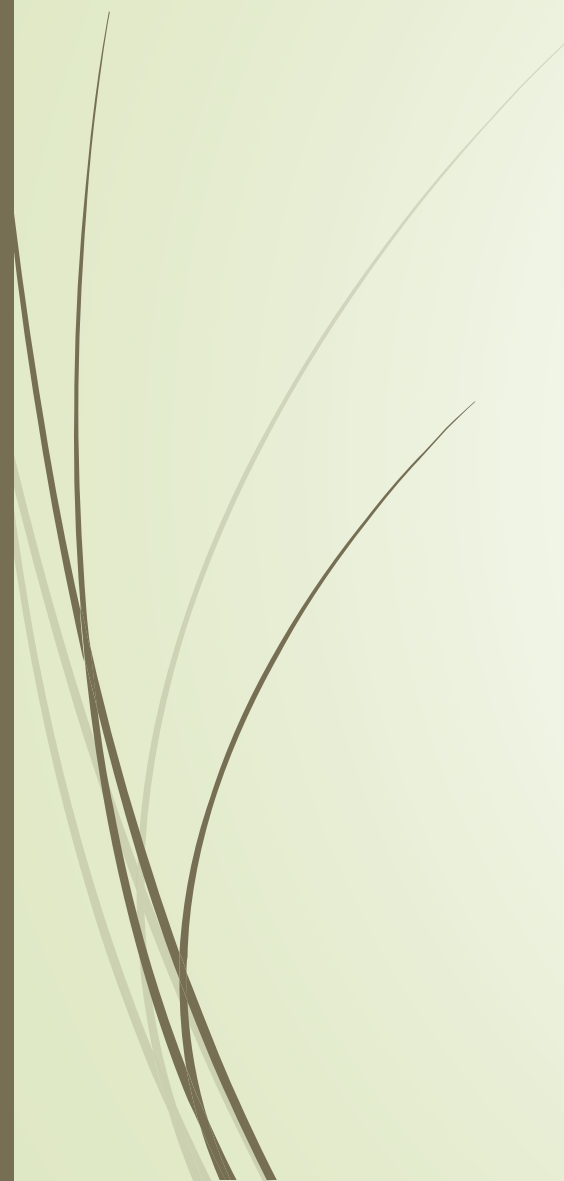
if {[info exists env{COMP_ENV}]} {
    puts stderr "\t[module-info name] Load Error: COMP_ENV env vars were not properly defined"
    puts stderr "\t[module-info name] is need a compiler"
    puts stderr "\tPlease set environment using the $ module load {compiler_module} [module-info name]"
    break
    exit 1
}

if {[info exists env{MPI_ENV}]} {
    puts stderr "\t[module-info name] Load Error: MPI_ENV env vars were not properly defined"
    puts stderr "\t[module-info name] is need a mpi library"
    puts stderr "\tPlease set environment using the $ module load {mpi_module} [module-info name]"
    break
    exit 1
}

#set prefix $prefix/$env{COMP_ENV}/$env{COMP_VER}/$env{MPI_ENV}/$env{MPI_VER}

if {[file exists $prefix]} {
    puts stderr "\t[module-info name] Load Error: [module-info name] doesn't support the loaded compiler or mpi library version"
    break
    exit 1
}

setenv      FFTWHOME      $prefix
setenv      DIFX_VERSION  DIFX-$version
prepend-path PATH          $prefix/bin
prepend-path CPATH         $prefix/include
prepend-path INCLUDE_PATH  $prefix/include
```

A solid red arrow pointing to the right, located in the upper left area of the slide.Several thin, dark brown curved lines that sweep upwards from the bottom left corner towards the center of the slide.

질의 응답

A red arrow graphic pointing to the right, located on the left side of the slide.

감사합니다.