

Programming Environments & Compilation



Starting at 11:05 am PST

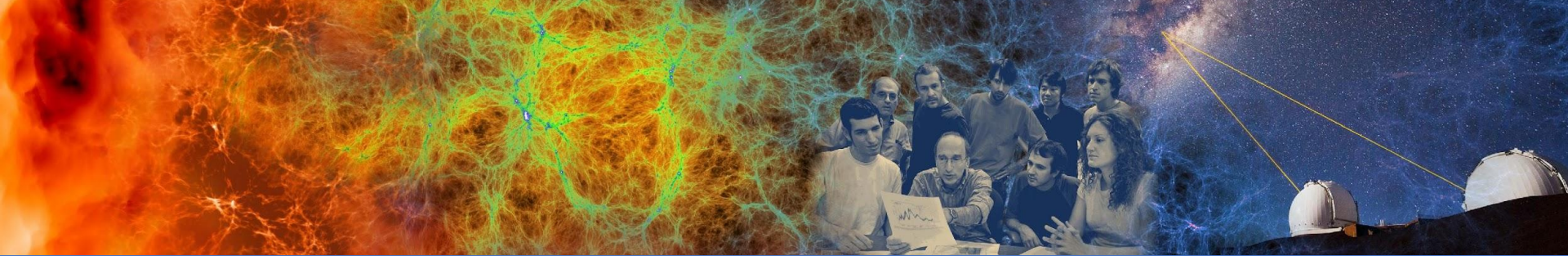
New User Training
February, 15 2023

Erik Palmer
Programming Environments and Models

Outline: Four Main Ways of Getting Software

- Loading modules
- Containers
- Compiling from source
- Conda, Spack & E4S

```
#####  
perlmutter  
#####  
Welcome to perlmutter!  
#####  
For all planned outages, see: https://www.nersc.gov/live  
For past outages, see: https://my.nersc.gov/outagelog-cs  
epalmer@perlmutter:login30:~> $  
epalmer@perlmutter:login30:~> $ Now what?  ^\_(\u03c9)\_/\u0304
```

Modules: Loading Preinstalled Software



BERKELEY LAB



U.S. DEPARTMENT OF
ENERGY

Office of
Science

Modules For Preinstalled Software: Ex. Python3

```
epalmer@perlmutter:login36:~> $ python --version
```

```
Python 2.7.18
```

```
epalmer@perlmutter:login36:~> $ module list
```

```
Currently Loaded Modules:
```

```
Currently Loaded Modules:
```

```
 1) craype-x86-milan      6) cray-dsmml/0.2.2      11) perftools-base/23.12.0
 2) libfabric/1.15.2.0    7) cray-libsci/23.12.5  12) cpe/23.12
 3) craype-network-ofi   8) cray-mpich/8.1.28    13) cudatoolkit/12.2
```

```
epalmer@perlmutter:login36:~> $ module load python/3.11
```

```
epalmer@perlmutter:login36:~> $ module list
```

```
Currently Loaded Modules:
```

```
 1) craype-x86-milan      6) cray-dsmml/0.2.2      11) perftools-base/23.12.0
 2) libfabric/1.15.2.0    7) cray-libsci/23.12.5  12) cpe/23.12
 3) craype-network-ofi   8) cray-mpich/8.1.28    13) cudatoolkit/12.2
```

```
14) python/3.11
```

```
(nersc-python) epalmer@perlmutter:login36:~> $ python --version
```

```
Python 3.11.7
```



176 Modules Currently Available on Perlmutter

Nsight-Compute: Nsight-Compute/2022.1.1
Nsight-Systems: Nsight-Systems/2022.2.1
OpenCoarrays: OpenCoarrays/2.10.1
PrgEnv-aocc:
PrgEnv-cray:
PrgEnv-gnu:
PrgEnv-intel:
PrgEnv-llvm: PrgEnv-llvm/0.1
PrgEnv-nvhpc:
PrgEnv-vidia:
R: R/4.2.3
amber:
aocc:
aocc-mixed:
arm-forge:
atp:
berkeleygw:
bupc: bupc/2022.10.0
bupc-narrow: bupc-narrow/2022.10.0
cce:
cce-mixed:
chapel: chapel/1.30.0
climate-utils:
cmake:
codee:
common-utils: common-utils/2023q2
conda:
contrib: contrib/1.0
cpe:
cpe-cuda:
cpu: cpu/1.0
cray-R:
cray-ccdb:
cray-cti:
cray-dsmdl: cray-dsmdl/0.2.2
cray-dyninst:
cray-fftw:
cray-hdf5:
cray-hdf5-parallel: cray-hdf5-parallel/1.12.2.9
cray-libpals:
cray-libsci:
cray-libsci_acc: cray-libsci_acc/23.12.0
cray-lustre-client-oved:
cray-mpich:

cray-mpich-abi:
cray-mpich-abi-pre-intel-5.0:
cray-mpich-ucx: cray-mpich-ucx/8.1.28
cray-mpich-ucx-abi: cray-mpich-ucx-abi/8.1.28
cray-mpixlate: cray-mpixlate/1.0.3
cray-mmet:
cray-netcdf:
cray-netcdf-hdf5parallel: cray-netcdf-hdf5parallel/4.9.0.9
cray-openshmemmx:
cray-pals:
cray-parallel-netcdf: cray-parallel-netcdf/1.12.3.9
cray-pmi:
cray-python:
cray-stat:
cray-ucx:
craype:
craype-accel-amd-gfx940: craype-accel-amd-gfx940
craype-accel-host: craype-accel-host
craype-accel-vidia70: craype-accel-vidia70
craype-accel-vidia80: craype-accel-vidia80
craype-accel-vidia90: craype-accel-vidia90
craype-hugepages128M: craype-hugepages128M
craype-hugepages16M: craype-hugepages16M
craype-hugepages1G: craype-hugepages1G
craype-hugepages256M: craype-hugepages256M
craype-hugepages2G: craype-hugepages2G
craype-hugepages2M: craype-hugepages2M
craype-hugepages32M: craype-hugepages32M
craype-hugepages4M: craype-hugepages4M
craype-hugepages512M: craype-hugepages512M
craype-hugepages64M: craype-hugepages64M
craype-hugepages8M: craype-hugepages8M
craype-network-none: craype-network-none
craype-network-ofi: craype-network-ofi
craype-network-ucx: craype-network-ucx
craype-x86-milan: craype-x86-milan
craype-x86-milan-x: craype-x86-milan-x
craype-x86-rome: craype-x86-rome
craypkg-gen:
cudatoolkit:
cudnn:
darshan:
dmtcp: dmtcp/3.0.0
dvs: dvs/2.15_4.5.239-2.5_48.29__g91483389

e4s:
eigen: eigen/3.4.0
espresso:
evp-patch: evp-patch
fast-mkl-amd: fast-mkl-amd/fast-mkl-amd
forge:
fpm: fpm/0.9.0
gcc:
gcc-mixed:
gcc-native: gcc-native/12.3
gcc-native-mixed: gcc-native-mixed/12.3
gdb4hpc:
globus-tools: globus-tools/1.0
gpu: gpu/1.0
gpu-test: gpu-test/1.1
gromacs:
gsi: gsi/2.7
hip:
idi:
impi: impi/2021.6.0
intel:
intel-classic:
intel-classic-mixed:
intel-llvm: intel-llvm/2023-WW13
intel-mixed:
intel-oneapi:
intel-oneapi-mixed:
iobuf: iobuf/2.0.10
jamo:
jgi: jgi/python-jamo
julia:
lammps: lammps/2022.11.03
libfabric: libfabric/1.15.2.0
libxc:
llvm:
lmod:
math-libs-real32: math-libs-real32/2023q2
mathematica: mathematica/13.0.1
matlab: matlab/r2021b
matlab-mcr: matlab-mcr/r2021b
mongodb:
mpich: mpich/4.1.1
mumps: mumps/5.5.1-gcc-11.2.0
mvasp: mvasp/5.4.4-cpu

namd:
nccl:
nersc_cr: nersc_cr/23.06
nvhpc:
nvhpc-mixed:
nvidia:
nvidia-mixed:
opencoarrays: opencoarrays/2.10.1
openmpi:
papi:
parallel: parallel/20210922
paraview: paraview/5.11.1
perftools: perftools
perftools-base:
perftools-lite: perftools-lite
perftools-lite-events: perftools-lite-events
perftools-lite-gpu: perftools-lite-gpu
perftools-lite-hbm: perftools-lite-hbm
perftools-lite-loops: perftools-lite-loops
perftools-preload: perftools-preload
petsc: petsc/3.19.3-cpu-complex
python:
pytorch:
qchem:
sanitizers4hpc:
settag: settag
spack:
spin: spin/2.0
taskfarmer: taskfarmer/1.5
tensorflow:
texlive: texlive/2022
totalview:
training: training/perlmutter-jan2022
upcxx:
upcxx-cuda:
upcxx-extras:
valgrind:
valgrind4hpc:
vasp:
vasp-tpc:
wannier90:
wps: wps/4.5.0
wrf: wrf/4.5.2
xpmem: xpmem/2.6.2-2.5_2.38__gd067c3f.shasta



Modules Loaded at Login

Modules Loaded by Default:

- | | | |
|--|------------------------|----------------------------|
| 1) craype-x86-milan | 6) cray-dsmml/0.2.2 | 11) perftools-base/23.12.0 |
| 2) libfabric/1.15.2.0 | 7) cray-libsci/23.12.5 | 12) cpe/23.12 |
| 3) craype-network-ofi | 8) cray-mpich/8.1.28 | 13) cudatoolkit/12.2 |
| 4) xpmem/2.6.2-2.5_2.38__gd067c3f.shasta | 9) craype/2.7.30 | 14) craype-accel-nvidia80 |
| 5) PrgEnv-gnu/8.5.0 | 10) gcc-native/12.3 | 15) gpu/1.0 |

- CPU Architecture
- Default Programming Environment and Compiler
- GPU Architecture and CUDA-Aware MPI

Modules with Lmod

Most Common

- `module list`
- `module load/unload`
- `module swap`
- `module show`
- `module spider`



Cool Tricks

- `module --redirect -r spider . | grep <string>`
- `ml -t`

More information: `man module` or <https://docs.nersc.gov/environment/lmod/>

Module Spider Example: Load cray-netcdf

No Longer
Recommended

- `module avail`

*Only shows packages that can be loaded into the current module environment (due to hierarchy) – **use `module spider` instead**

```
epalmer@perlmutter:login34:~> $
```

More information: `man module` or <https://docs.nersc.gov/environment/lmod/>

Module Spider Example: Load cray-netcdf

No Longer
Recommended

- `module avail`

*Only shows packages that can be loaded into the current module environment (due to hierarchy) – **use `module spider` instead**

```
epalmer@perlmutter:login34:~> $
```

More information: `man module` or <https://docs.nersc.gov/environment/lmod/>

Loading Modules Modifies Your Environment

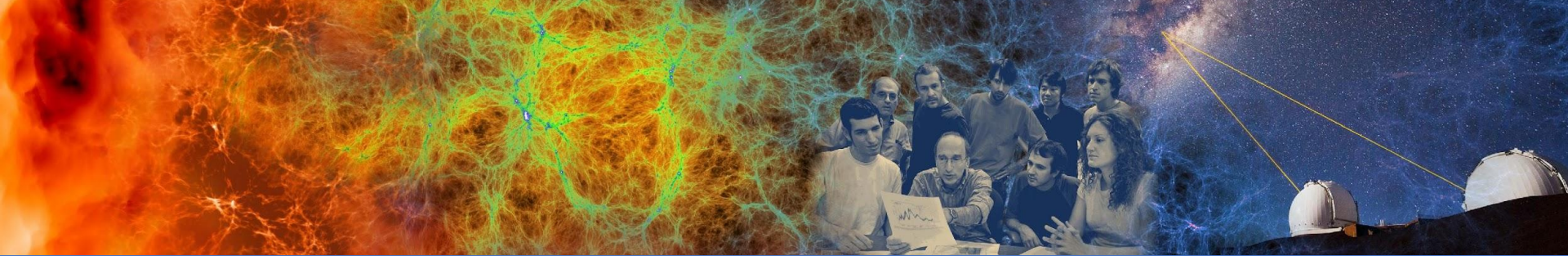
```
epalmer@perlmutter:login25:~/Training> $ module show cray-hdf5
```

```
-----  
/opt/cray/pe/lmod/modulefiles/compiler/gnu/12.0/cray-hdf5/1.12.2.9.lua:  
-----
```

```
family("hdf5")  
conflict("PrgEnv-pathscale")  
conflict("cray-hdf5")  
conflict("cray-hdf5-parallel")  
help([[Release info: /opt/cray/pe/hdf5/1.12.2.9/release_info]])  
whatis("The HDF5 Technology suite includes tools and applications for managing, manipulating, viewing, and analyzing  
prepending PATH, /opt/cray/pe/hdf5/1.12.2.9/bin")  
prepending PKG_CONFIG_PATH, /opt/cray/pe/hdf5/1.12.2.9/gnu/12.3/lib/pkgconfig")  
prepending PE_PKGCONFIG_LIBS, hdf5_hl:hdf5")  
setting PE_HDF5_PKGCONFIG_LIBS, hdf5_hl:hdf5")  
prepending PE_FORTRAN_PKGCONFIG_LIBS, hdf5hl_fortran:hdf5_fortran")  
setting PE_HDF5_FORTRAN_PKGCONFIG_LIBS, hdf5hl_fortran:hdf5_fortran")  
prepending PE_CXX_PKGCONFIG_LIBS, hdf5_hl_cpp:hdf5_cpp")  
setting PE_HDF5_CXX_PKGCONFIG_LIBS, hdf5_hl_cpp:hdf5_cpp")  
setting CRAY_HDF5_DIR, /opt/cray/pe/hdf5/1.12.2.9")  
setting PE_HDF5_DIR, /opt/cray/pe/hdf5/1.12.2.9")  
setting CRAY_HDF5_VERSION, 1.12.2.9")  
setting CRAY_HDF5_PREFIX, /opt/cray/pe/hdf5/1.12.2.9/gnu/12.3")  
setting HDF5_DIR, /opt/cray/pe/hdf5/1.12.2.9/gnu/12.3")  
setting HDF5_ROOT, /opt/cray/pe/hdf5/1.12.2.9/gnu/12.3")  
prepending CRAY_LD_LIBRARY_PATH, /opt/cray/pe/hdf5/1.12.2.9/gnu/12.3/lib")  
prepending MODULEPATH, /opt/cray/pe/lmod/modulefiles/hdf5/gnu/12.0/cray-hdf5/1.12.2")
```



Path Changes
Environment Variables
Other Info



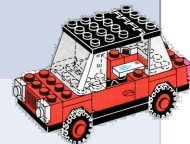
Programming Environments: Configuring Compilers and Libraries

Compiling: Your Machine vs. Perlmutter

What you may have done on another machine:

```
epalmer@home_machine:~> gcc helloworld.c -o helloworld.ex
```

- Not MPI-enabled code.



```
epalmer@home_cluster:~> mpicc helloworld.c -o helloworld.ex
```

- MPI compiler wrapper includes MPI libraries



What we recommend for Perlmutter:

```
epalmer@perlmutter:~> module load PrgEnv-gnu  
epalmer@perlmutter:~> cc helloworld.c -o helloworld.ex
```

- HPE compiler wrapper includes MPI libraries, optimizations and more.



Compiler Wrappers and a Useful Option

-v and -craype-verbose will show all the inputs added by the craype (Cray Programming Environment) to the compiler when the wrappers (CC, cc, ftn) are used.

```
epalmer@nid005015:~/Training> CC -craype-verbose helloworld.cpp -o helloworld.ex
```

```
g++-12 -march=znver3 -D__CRAY_X86_MILAN -D__CRAY_NVIDIA80  
-D__CRAYXT_COMPUTE_LINUX_TARGET -D__TARGET_LINUX__ helloworld.cpp -o  
helloworld.ex -Wl,-Bdynamic  
-I/opt/cray/pe/mpich/8.1.28/ofc/gnu/12.3/include  
-I/opt/cray/pe/libsci/23.12.5/GNU/12.3/x86_64/include  
-I/opt/nvidia/hpc_sdk/Linux_x86_64/23.9/cuda/12.2/nvvm/include  
-I/opt/nvidia/hpc_sdk/Linux_x86_64/23.9/cuda/12.2/extras/CUPTI/include  
-I/opt/nvidia/hpc_sdk/Linux_x86_64/23.9/cuda/12.2/extras/Debugger/include ...(and more)
```

Compiler Wrapper Includes A Lot

```
epalmer@nid005015:~/Training> CC -craype-verbose helloworld.cpp -o hello
```

```
g++-12 -march=znver3 -D__CRAY_X86_MILAN -D__CRAY_NVIDIA80 -D__CRAYXT_COMPUTE_LINUX_TARGET  
-D__TARGET_LINUX__ helloworld.cpp -o helloworld.ex -Wl,-Bdynamic  
-I/opt/cray/pe/mpich/8.1.28/ofl/gnu/12.3/include  
-I/opt/cray/pe/libsci/23.12.5/GNU/12.3/x86_64/include  
-I/opt/nvidia/hpc_sdk/Linux_x86_64/23.9/cuda/12.2/nvvm/include  
-I/opt/nvidia/hpc_sdk/Linux_x86_64/23.9/cuda/12.2/extras/CUPTI/include  
-I/opt/nvidia/hpc_sdk/Linux_x86_64/23.9/cuda/12.2/extras/Debugger/include -I/opt/cray/pe/dsmml/0.2.2/dsmml//include  
-I/opt/cray/xpmem/2.6.2-2.5_2.38__gd067c3f.shasta/include  
-L/opt/cray/pe/mpich/8.1.28/ofl/gnu/12.3/lib -L/opt/cray/pe/mpich/8.1.28/gtl/lib -L/opt/cray/pe/libsci/23.12.5/GNU/12.3/x86_64/lib  
-L/opt/nvidia/hpc_sdk/Linux_x86_64/23.9/cuda/12.2/lib64/stubs -L/opt/nvidia/hpc_sdk/Linux_x86_64/23.9/cuda/12.2/lib64  
-L/opt/nvidia/hpc_sdk/Linux_x86_64/23.9/cuda/12.2/nvvm/lib64  
-L/opt/nvidia/hpc_sdk/Linux_x86_64/23.9/cuda/12.2/extras/CUPTI/lib64  
-L/opt/nvidia/hpc_sdk/Linux_x86_64/23.9/cuda/12.2/extras/Debugger/lib64  
-L/opt/nvidia/hpc_sdk/Linux_x86_64/23.9/math_libs/12.2/lib64 -L/opt/cray/pe/dsmml/0.2.2/dsmml//lib  
-L/opt/cray/xpmem/2.6.2-2.5_2.38__gd067c3f.shasta/lib64 -Wl,--as-needed,-lcupti,-lcudart,--no-as-needed -lcuda  
-Wl,--as-needed,-lsci_gnu_123_mpi,--no-as-needed -Wl,--as-needed,-lmpi_gnu_123,--no-as-needed -lmpi_gtl_cuda  
-Wl,--as-needed,-lsci_gnu_123,--no-as-needed -ldl -Wl,--as-needed,-ldsmml,--no-as-needed -lxpmem  
-Wl,--as-needed,-lgfortran,-lquadmath,--no-as-needed -Wl,--as-needed,-lmvec,--no-as-needed -Wl,--as-needed,-lm,--no-as-needed  
-Wl,--as-needed,-lthread,--no-as-needed -Wl,--disable-new-dtags
```

Same Wrapper, Different Compiler

```
epalmer@perlmutter:~> module load PrgEnv-nvidia
```

```
epalmer@perlmutter:~> CC -craype-verbose helloworld.cpp -o helloworld.ex
```

```
nvc++ -tp=zen3 -acc -gpu=cc80 -D__CRAY_X86_MILAN -D__CRAY_NVIDIA80
```

```
-D__CRAYXT_COMPUTE_LINUX_TARGET -pgf90libs helloworld.cpp -o helloworld.ex
```

```
-l/opt/cray/pe/mpich/8.1.28/ofc/nvidia/23.3/include ...
```

```
epalmer@perlmutter:~> module load PrgEnv-intel
```

```
epalmer@perlmutter:~> CC -craype-verbose helloworld.cpp -o helloworld.ex
```

```
icpx -march=core-avx2 -mtune=core-avx2 -D__CRAY_X86_MILAN -D__CRAY_NVIDIA80
```

```
-D__CRAYXT_COMPUTE_LINUX_TARGET helloworld.cpp -o helloworld.ex
```

```
-Wl,-rpath=/opt/intel/oneapi/compiler/2023.2.0/linux/compiler/lib/intel64
```

```
-l/opt/cray/pe/mpich/8.1.28/ofc/intel/2022.1/include ...
```

PrgEnvs, Compilers and Libraries

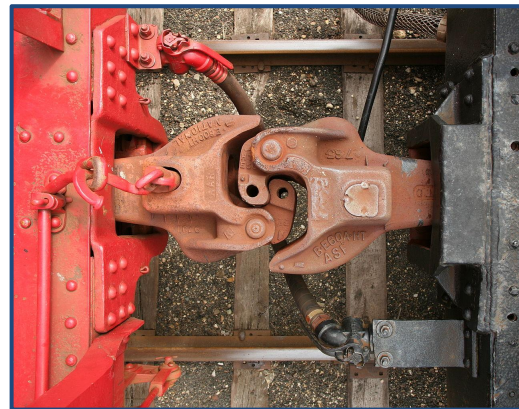
Module	Compiler	C++		C		Fortran		MPI Library
PrgEnv-gnu	GNU	CC	g++	cc	gcc	ftn	gfortran	cray-mpich
PrgEnv-nvidia	Nvidia HPC	CC	nvc++	cc	nvc	ftn	nvfortran	cray-mpich
PrgEnv-intel	Intel	CC	icpx	cc	icx	ftn	ifort	cray-mpich
PrgEnv-cray	Cray	CC	crayCC (Clang)	cc	craycc (Clang)	ftn	crayftn	cray-mpich

For additional compilers and more info: <https://docs.nersc.gov/development/compilers/base/>

Automatic Links Provided By The Wrappers

- Depending on modules loaded, compiler wrappers link:

MPI, LAPACK, Blas, ScaLAPACK,
and more, **automatically**.



- Cray modules, such as `cray-hdf5`, `cray-fftw`, etc. are also linked automatically by the compiler wrappers when loaded into the user environment.

Note: Several scientific libraries such as, LAPACK, ScaLAPACK, Blas and QDWH, are included in `cray-libsci`. For more information use: `man intro_libsci`.

Build Systems: Autoconf and CMake

Build systems such as CMake or Autotools (Makefiles) may be coded to search for the environment variables:

CC, for the C compiler ;
CXX, for the C++ compiler;
FC, for the Fortran compiler .

In this case, the Cray compile wrappers can be specified with:

```
CC=$(which cc) CXX=$(which CC) FC=$(which ftn)
```

Or at the configure step,

```
./configure CC=cc CXX=CC FC=ftn F77=ftn
```

More info: <https://docs.nersc.gov/development/build-tools/autoconf-make/>
<https://docs.nersc.gov/development/build-tools/cmake/>

Build Systems Example: Compiling SLATE

From INSTALL.md:

```
Configure and compile the SLATE library and its tester,  
then install the headers and library. This will also compile  
BLAS++, LAPACK++, and TestSweeper.
```

```
**Option 1: Makefile**
```

```
# create make.inc file, for example:  
CXX = mpicxx      # MPI compiler wrappers recommended  
FC   = mpif90  
blas = openblas  
CXXFLAGS = -DSLATE_HAVE_MT_BCAST
```

```
Compile and install:
```

```
make && make install
```

This build will require specifying:

CC=cc CXX=CC FC=ftn

Build Systems Tip: CMake's GUI, ccmake

```
Page 1 of 3
CMAKE_ADDR2LINE      * /usr/bin/addr2line
CMAKE_AR             * /usr/bin/ar
CMAKE_BUILD_TYPE     *
CMAKE_COLOR_MAKEFILE * ON
CMAKE_CRAYPE_LINKTYPE * dynamic
CMAKE_CRAYPE_LOADEDMODULES * craype-x86-milan:libfabric/1.15.2.0:craype-network-
CMAKE_CXX_COMPILER   * /opt/cray/pe/craype/2.7.30/bin/CC
CMAKE_CXX_COMPILER_AR * /usr/bin/gcc-ar-12
CMAKE_CXX_COMPILER_RANLIB * /usr/bin/gcc-ranlib-12
CMAKE_CXX_FLAGS     *
CMAKE_CXX_FLAGS_DEBUG * -g
CMAKE_CXX_FLAGS_MINSIZEREL * -Os -DNDEBUG
CMAKE_CXX_FLAGS_RELEASE * -O3 -DNDEBUG
CMAKE_CXX_FLAGS_RELWITHDEBINFO * -O2 -g -DNDEBUG
CMAKE_C_COMPILER     * /opt/cray/pe/craype/2.7.30/bin/cc
CMAKE_C_COMPILER_AR * /usr/bin/gcc-ar-12
CMAKE_C_COMPILER_RANLIB * /usr/bin/gcc-ranlib-12
CMAKE_C_FLAGS       *
CMAKE_C_FLAGS_DEBUG * -g

CMAKE_ADDR2LINE: Path to a program.
Keys: [enter] Edit an entry [d] Delete an entry
      [l] Show log output  [c] Configure
      [h] Help             [q] Quit without generating
      [t] Toggle advanced mode (currently on)
```

In the CMake build directory, type:
ccmake ..

Can confirm the correct compiler wrappers are being used by the build system

Two Comments about Linking:

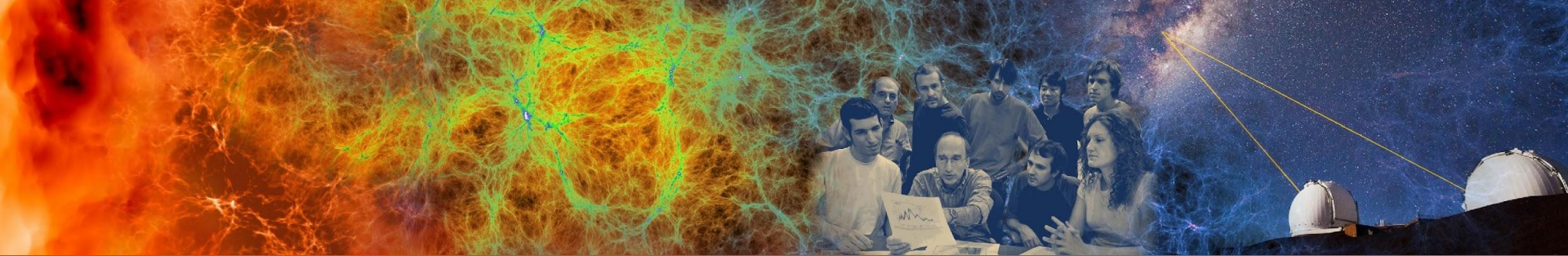
- Many modules prepend the `LIBRARY_PATH` so it is not necessary to specify library locations, e.g.

```
elvis@login01:~> $ module load gsl
elvis@login01:~> $ CC gsl_test.cpp -lgsl -lgslcblas -o gsl_test
```

Dynamic vs. Static Linking

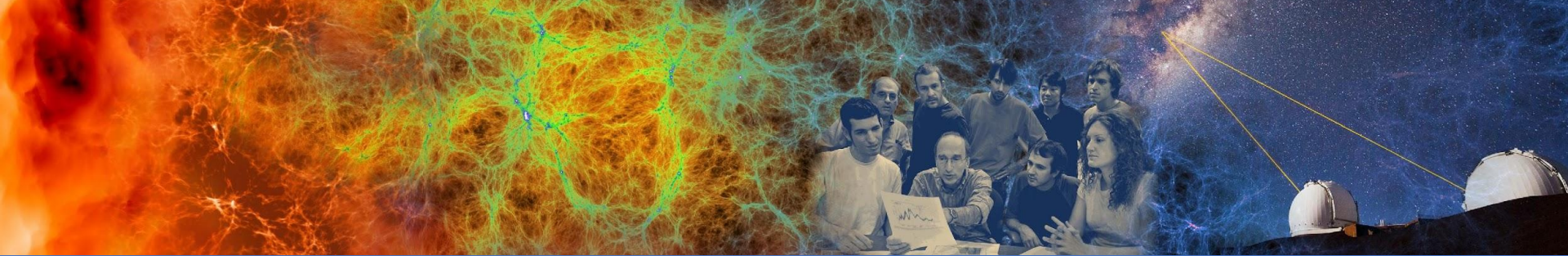
- Cray wrappers build dynamically linked executables by default. (The kind that give, “error while loading shared libraries: ...” when they get misplaced at runtime.)
- On Perlmutter static compilation with `-static` or `CRAYPE_LINK_TYPE=static` can fail and is not supported.

More info: <https://docs.nersc.gov/development/compilers/wrappers/>



Best Practice For Compiling Code on Perlmutter:

- Use the system compiler wrappers, `cc`, `CC`, and `ftn`
- With build systems, verify the compiler wrappers are being used



Examples of Compiling Code

Example CPU Code Compile with MPI and OpenMP

```
int main(int argc, char *argv[]) {
    int numprocs, rank, namelen;
    char processor_name[MPI_MAX_PROCESSOR_NAME];
    int iam = 0, np = 1;
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Get_processor_name(processor_name, &namelen);

    #pragma omp parallel default(shared) private(iam, np)
    {
        np = omp_get_num_threads();
        iam = omp_get_thread_num();
        printf("Hello from thread %d out of %d from process
              %d out of %d on %s\n",
              iam, np, rank, numprocs, processor_name);
    }
    MPI_Finalize();
}
```

Hellohybrid.c – contains both MPI and OpenMP commands.

Example from:

<https://rcc.uchicago.edu/docs/runnning-jobs/hybrid/index.html>

Some Good-to-Know Compiler Settings

GNU	Cray	Nvidia	Description/ Comment
-O0	-O0	-O1	Default Optimization Level
-Ofast	-Ofast, -flto	-O4, -fast	Aggressive Optimization (some may cause non-bit-identical output)
-fopenmp	-fopenmp	-mp (CPU), -mp=gpu (GPU offload)	Enable OpenMP (not default)
		-acc	Enable OpenACC
-g, -O0	-g, -O0	-g (-O0 by default)	Debug
-v	-v	-v	Verbose

For more information:

man gcc/gfortran

man craycc/crayftn

man nvc/nvfortran

<https://docs.nersc.gov/development/compilers/>

Compile with MPI and OpenMP (CPU only)

```
epalmer@nid006368:~/NERSC_User_Training/EX1> #In this example, we will compile a  
code with MPI and OpenMP
```

Modules Loaded:
PrgEnv-gnu

Compile line: `cc hellohybrid.c -fopenmp -o hellohybrid`

Compile with MPI and OpenMP (CPU only)

```
epalmer@nid006368:~/NERSC_User_Training/EX1> █
```

Modules Loaded:
PrgEnv-gnu

Compile line: `cc hellohybrid.c -fopenmp -o hellohybrid`

Compile with MPI and OpenMP on GPU

```
epalmer@nid003676:~/NERSC_User_Training/EX1> ls
hellohybrid.c
epalmer@nid003676:~/NERSC_User_Training/EX1> module list

Currently Loaded Modules:
  1) craype-x86-milan          10) cudatoolkit/11.7
  2) libfabric/1.15.0.0      11) craype-accel-nvidia80
  3) craype-network-ofi     12) gpu/1.0
  4) perftools-base/22.06.0 13) nvidia/22.5
  5) xpmem/2.4.4-2.3_12.2__gff0e1d9.shasta 14) craype/2.7.16
  6) xalt/2.10.2            15) cray-dsmml/0.2.2
  7) darshan/3.3.1         16) cray-mpich/8.1.17
  8) Nsight-Compute/2022.1.1 17) cray-libsci/21.08.1.2
  9) Nsight-Systems/2022.2.1 18) PrgEnv-nvidia/8.3.3

epalmer@nid003676:~/NERSC_User_Training/EX1> cc
```

Modules Loaded:
PrgEnv-nvidia,
gpu

“-Minfo” is an optional command shows which parts were converted to NVIDIA GPU code.

Compile line: `cc hellohybrid.c -mp=gpu -Minfo -o hellohybrid`

Compile with MPI and OpenMP on GPU

```
epalmer@nid003676:~/NERSC_User_Training/EX1> █
```

Modules Loaded:
PrgEnv-nvidia,
gpu

“-Minfo” is an optional command shows which parts were converted to NVIDIA GPU code.

```
Compile line: cc hellohybrid.c -mp=gpu -Minfo -o hellohybrid
```

CUDA-Aware MPI

Modules Loaded by Default:

1) craype-x86-milan	7) cray-libsci/23.12.5	13) cudatoolkit/12.2
2) libfabric/1.15.2.0	8) cray-mpich/8.1.28	14) craype-accel-nvidia80
3) craype-network-ofi	9) craype/2.7.30	15) gpu/1.0
4) xpmem/2.6.2-2.5_2.38__gd067c3f.shasta	10) gcc-native/12.3	
5) PrgEnv-gnu/8.5.0	11) perftools-base/23.12.0	
6) cray-dsmml/0.2.2	12) cpe/23.12	

Output of module show gpu:

```
-----  
/opt/nersc/pe/modulefiles/gpu/1.0.lua:  
-----  
family("hardware")  
load("cudatoolkit")  
load("craype-accel-nvidia80")  
setenv("MPICH_GPU_SUPPORT_ENABLED", "1")
```

Note: The gpu module enables support for CUDA-Aware MPI – Allowing MPI to copy data to and from GPUs. module load cpu will turn it off.

Compile a CUDA Code with CUDA-Aware MPI

```
epalmer> # In this example, we will compile a CUDA code with CUDA-aware MPI
epalmer> ls
kernels.cu  kernels.h  vecAdd.cpp
epalmer> module list
```

Modules Loaded:
PrgEnv-nvidia,
gpu

Note: The flag, `MPICH_GPU_SUPPORT_ENABLED` turns CUDA-Aware MPI on or off. Loading the `gpu` module, sets this flag to 1, thereby enabling the feature.

Compile a CUDA Code with CUDA-Aware MPI

```
epalmer> █
```

Modules Loaded:
PrgEnv-nvidia,
gpu

Note: The flag, `MPICH_GPU_SUPPORT_ENABLED` turns CUDA-Aware MPI on or off. Loading the `gpu` module, sets this flag to 1, thereby enabling the feature.

Compile Commands for Code with OpenACC

Modules Loaded:

PrgEnv-nvidia,
gpu

Compile with OpenACC enabled:

```
cc helloacc.c -acc -Minfo=acc -o helloacc
```

*Optional command shows which parts were converted to NVIDIA GPU code.

Manually Specify Include, Library Location and Links

```
epalmer> # In this example, we will show how to manually include and link libraries during  
the compile step. The example code I will use, requires 
```


Manually Specify Include, Library Location and Links

```
epalmer> []
```

More Resources on Compiling Code

NERSC Docs:

- Compiling a code on NERSC resources: 

<https://docs.nersc.gov/tutorials/playbooks/compiling/>

- Compiling and Building Software:

<https://docs.nersc.gov/systems/perlmutter/#compilingbuilding-software>

- Base Compilers:

<https://docs.nersc.gov/development/compilers/base/#base-compilers-on-nersc-systems>

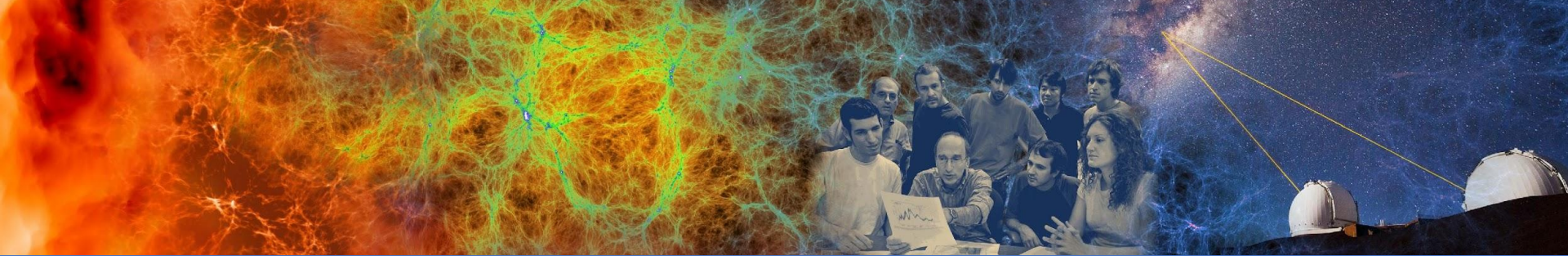
- Compiler Wrappers:

<https://docs.nersc.gov/development/compilers/wrappers/#compiler-wrappers>

Supported Programming Models



and more at <https://docs.nersc.gov/development/programming-models/>



Where to install your code



BERKELEY LAB



U.S. DEPARTMENT OF
ENERGY

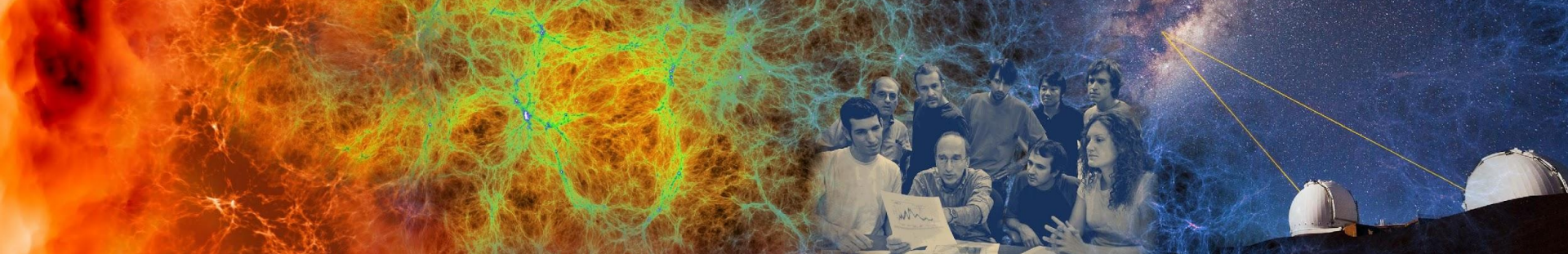
Office of
Science

Possible Install Locations

Note: Common default install locations such as /usr are not writable for Perlmutter users.

Location	Directory	Performance	Notes
Home	\$HOME	nonoptimal	Source code, scripts, small
Scratch	\$SCRATCH	optimal	Unsharable, for temporary I/O
Common	/global/common/ software/<proj>	performant	Optimized for software installs, Read-only on compute nodes
Community	\$CFS/<proj>	medium	For sharing data among projects

More info: <https://docs.nersc.gov/filesystems/>



Spack: A Package Manager for Supercomputers

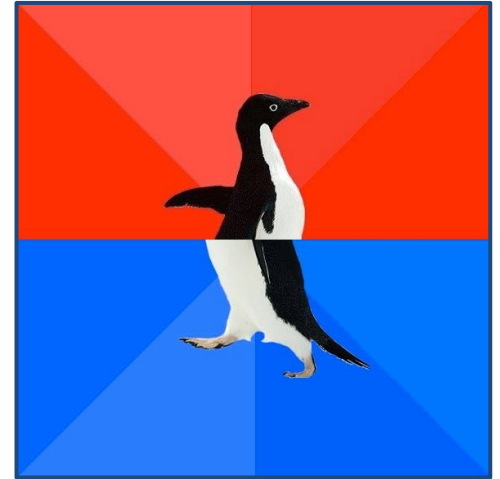
Installing a package with Spack

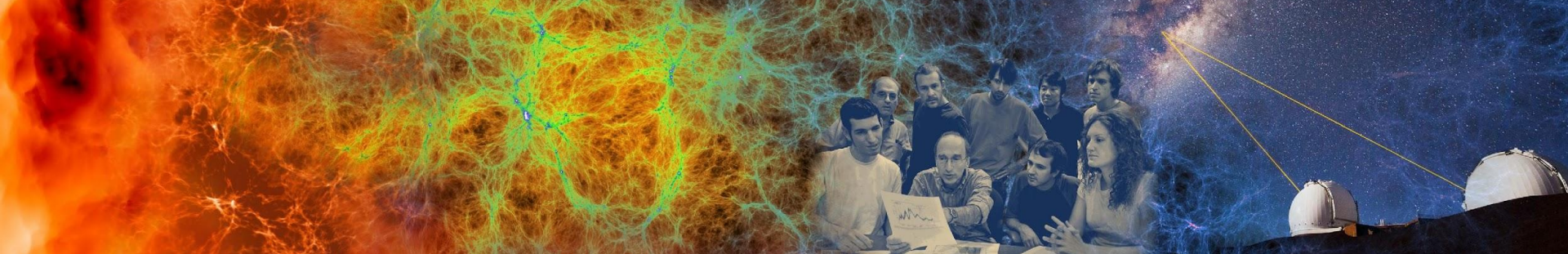


```
git clone https://github.com/spack/spack.git  
. spack/share/spack/setup-env.sh  
spack install <package>
```

- Typically installs take some time as Spack will download and build many redundant dependencies.
- You can tell Spack not to do this with configuration files.

Since Spack is widely used, there is a lot of support. See <https://spack.io/about/> for more information and check out their Slack channel.





E4S

Extreme-Scale Scientific Software Stack Even More Software!



BERKELEY LAB



U.S. DEPARTMENT OF
ENERGY

Office of
Science

The Extreme-Scale Scientific Software Stack (E4S)

module	Version	Command	Installed Packages	Can I Install More?
e4s	23.08	module load e4s	478	Yes

```
elvis@perlmutter> module load e4s
elvis@perlmutter> spack env activate cuda
elvis@perlmutter> spack find
==> In environment cuda
==> Root specs
-- no arch / gcc@12.2.0
-----
amrex%gcc@12.2.0 +cuda cuda_arch=80
arborx%gcc@12.2.0 +cuda cuda_arch=80
caliper%gcc@12.2.0 +cuda cuda_arch=80
...
```

- E4S at NERSC is a curated software stack that get additional testing and support – delivered via the SPACK package manager

More info:

<https://docs.nersc.gov/applications/e4s/>

Pre-Installed Spack Packages

Spack

```

binutils@2.40
bitgroomingz@2022-10-14
blaspp@2022.07.00
blt@0.5.2
bolt@2.0
boost@1.82.0
boost@1.82.0
boost@1.82.0
boost@1.82.0
boost@1.82.0
boost@1.82.0
boost@1.82.0
boost@1.82.0
bzip2@1.0.6
c-blosc@1.21.2
cabana@0.5.0
cairo@1.16.0
caliper@2.9.0
camp@2022.10.1
camp@2022.10.1
cdo@2.1.1
chai@2022.03.0
chapel@1.24.1
cmake@3.26.3
cpio@2.13
cray-libsci@0s
cray-mpich@0s
curl@7.66.0
datatransferkit@3.1-rc3
diffutils@3.6
dynaminst@12.3.0
dyninst@12.3.0
eccodes@2.25.0
eigen@3.4.0
elfutils@0.189
elfutils@0.189
esmf@8.4.2
exmcutils@0.6.0
expat@2.5.0
fft
gotcha@1.0.4
gotcha@1.0.4
gperftools@2.10
gromacs@2023.1
gslib@2.7.1
h5bench@1.3
hdf5@1.14.1-2
hdf5@1.14.1-2
hdf5@1.14.1-2
hdf5@1.14.1-2
hdf5-vol-async@1.5
hdf5-vol-cache@v1.1
heffte@2.3.0
hpctoolkit@2023.03.01
hpcviewer@2023.04
hpx@1.9.0
hwloc@2.9.1
hypre@2.28.0
inputproto@2.3.2
intel-mkl@2020.4.304
intel-tbb@2020.3
intel-tbb@2020.3
intel-xed@2022.10.11
jansson@2.14
json-c@0.16
kbproto@1.0.7
kokkos@3.7.00
kokkos@4.0.01
kokkos@4.0.01
kokkos-kernels@3.7.00
lakkos@20220623.3
axlib@0.7.1
bacio@2.4.1
bdftopcf@1.0.5
binutils@2.38
bison@3.8.2
blaspp@2022.07.00
blaspp@2022.07.00
blt@0.5.2
blt@0.5.2
blt@0.5.2
boost@1.80.0
boost@1.80.0
butterflypack@2.2.2
butterflypack@2.2.2
bzip2@1.0.6
c-blosc@1.21.1
c-blosc@1.21.1
cabana@0.5.0
libstdcompat@0.0.17
libtiff@4.5.0
libunwind@1.6.2
libunwind@1.6.2
libhuu@1.44.1
epalmer@perlmutter:login31:-> $ spack -E find
-- linux-sles15-zen3 / gcc@11.2.0
-----
freetype@2.11.1
frabidi@1.0.12
g2@3.4.5
gdk-pixbuf@2.42.9
gettext@0.21.1
gettext@0.21.1
ghostscript@9.56.1
ginkgo@1.4.0
ginkgo@1.4.0
git@2.38.1
glib@2.74.1
glib@2.74.1
globalarrays@5.8
glproto@1.4.17
glx@1.4
gmake@4.3
gmake@4.3
gmp@6.2.1
gnutls@3.7.8
gobject-introspection@1.72.0
gobject-introspection@1.72.0
googletest@1.8.1
gotcha@1.0.4
gperftools@3.1
grace@5.1.25
grads@2.2.1
graphviz@2.1.3.14
grib-util@11.2.4
gslib@2.7.1
gtk-doc@1.33.2
gtkplus@3.24.29
h5bench@1.3
hdf5@2.38.1
hdf5@1.12.2
hdf5@1.12.2
hdf5@1.12.2
heffte@2.3.0
hwp2man@1.47.16
hwloc@2.8.0
hwloc@2.8.0
hwloc@2.8.0
hwloc@2.8.0
libxft@1.5
libxft@1.2.2
libyaml@0.2.5
papyrus@1.0.2
parallelio@2.5.10
parmetis@04.0.3
parsec@3.0.2209
pcrere@0.45
perl@5.34.0
perl@5.34.0
py-nbclient@0.7.2
py-nbconvert@7.0.0
py-nbformat@5.8.0
py-nest-asyncio@1.5.6
py-notebook@6.4.12
stc@0.9.0
strumpack@7.1.1
sundials@6.5.1
sundials@6.5.1
superlu@5.3.0
superlu-dist@7.2.0
libtirpc@1.3.3
libtirpc@1.3.3
libtool@2.4.7
libtool@2.4.7
libunistring@0.9.10
libunwind@1.6.2
libuv@1.44.1
libwebp@1.2.2
libx11@1.7.0
libx11@1.7.0
libxa@1.0.8
libxaw@1.0.13
libxaw@1.0.13
libxcb@1.14
libxcb@1.14
libxcomposite@0.4.4
libxcomposite@0.4.4
libxdmcp@1.1.2
libxext@1.3.3
libxext@1.3.3
libxf86@0.5.0.2
libxf86@0.5.0.2
libxft@2.3.2
libxft@1.7.6
libxkbcommon@1.4.0
libxkbcommon@1.4.0
libxkbfile@1.0.9
libxml2@2.10.1
libxml2@2.10.1
libxmu@1.1.2
libxmu@1.1.2
libxp@1.0.3
libxpm@3.5.12
libxpm@3.5.12
libxrandr@1.5.0
libxrender@0.9.10
libxrender@0.9.10
libxslt@1.1.33
libxslt@1.1.33
libxtrans@0.40.0
libxt@1.1.5
libxt@1.1.5
libxtst@1.2.2
libyaml@0.2.5
parmetis@4.0.3
parsec@3.0.2209
pcrere@0.45
pcrere@10.39
pdsh@2.31
pdt@3.25.1
perl@5.26.1
perl-data-dumper@2.173
perl-encode-locale@1.05
perl-extutils-config@0.008
perl-extutils-helpers@0.026
perl-extutils-installpaths@0.012
perl-file-listing@0.04
perl-html-parser@0.72
perl-html-tagset@3.20
perl-http-cookies@0.04
perl-http-daemon@0.01
perl-http-date@0.02
perl-http-message@0.13
perl-http-negotiate@0.06
perl-io-html@1.001
perl-libwww-perl@6.33
perl-lwp-mediatypes@0.02
perl-module-build@0.4224
perl-module-build-tiny@0.039
perl-net-http@0.17
perl-test-needs@0.02005
perl-try-tiny@0.28
perl-uri@1.72
perl-www-robotrules@0.02
perl-xml-parser@2.44
petsc@3.18.1
petsc@3.18.1
petsc@3.18.1
petsc@3.18.1
pflotran@0.4.0.1
pfnunit@0.9.10
pfnunit@0.9.10
phist@1.11.2
pig@2.7
pixman@0.40.0
pkg-config@0.29.2
plumed@2.8.0
poppler@0.79.0
python@3.6.15
python@3.9.7
python@3.9.7
qt@5.15.5
qt@5.15.5
qthreads@1.16
quantum-espresso@7.1
raja@2022.03.0
raja@2022.03.0
randrproto@1.5.0
rankstr@0.1.0
readline@8.1.2
recordproto@1.14.2
redset@0.2.0
renderproto@0.11.1
rsh@1.4.2
rust@1.60.0
scotch@0.1
scr@3.0.1
sed@4.8
shapelib@1.5.0
shared-mime-info@1.9
shuffrle@0.2.0
slate@2022.07.00
slate@2022.07.00
slepc@3.18.1
slepc@3.18.1
slepc@3.18.1
slepc@3.18.1
slepc@3.18.1
slurm@2022.05.8
snappy@1.1.9
snappy@1.1.9
sp@2.3.3
sp@2.3.3
sp@2.3.3
sqlite@3.39.4
stc@0.9.0
strumpack@7.0.1
strumpack@7.0.1
suite-sparse@5.13.0
sundials@6.4.1
sundials@6.4.1
superlu@5.3.0
superlu-dist@0.1.1

```

E4S



Install or Load a Spack Package

Steps	Already Installed in Spack/E4S	Available via Spack/E4S
1	<code>spack env activate <env></code>	
	Select an environment. *Needed for E4S modules, not required for Spack module	
2	<code>spack find -v</code>	<code>spack list</code>
	Match package config. and compiler	Search list of ~6700 avail. packages
3	<code>spack load <package></code>	<code>spack info <package></code>
	Load desired package/variant	Get information about package options
4		<code>spack install <package></code>
		Installs the package into Spack
5		<code>spack load <package></code>

Bonus: `spack load --sh <package>`

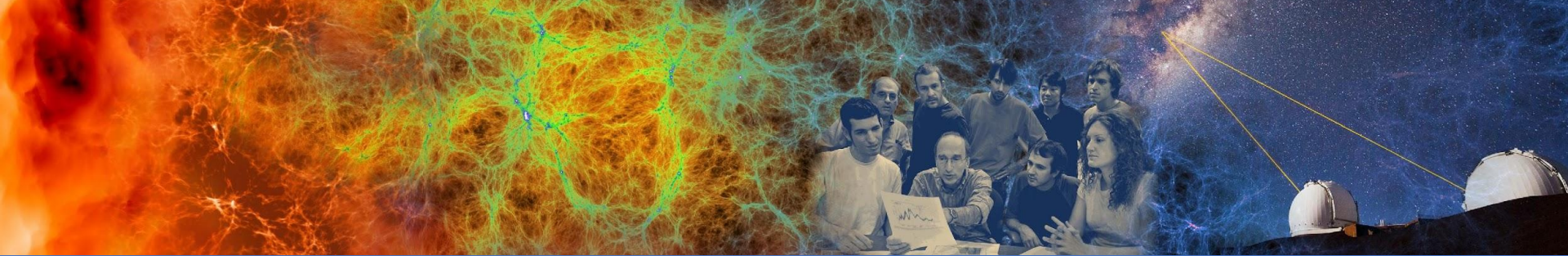
Show modifications made to your environment when a package is loaded.

Example: Install a Dependency with Spack

```
+phalana+piro+rol+ryttaa+saacae+shanda+shyly+stak+strakho+stratimikos+superia-di
st+teko+tepus+tperra+trifinoscouplings+zoltan+zoltan2_gotype=long_long
umap@2.1.0%gcc@11.2.0
umpire@2022.03.1%gcc@11.2.0
upcxx@2022.0.0%gcc@11.2.0
vtk-m@1.9.0%gcc@11.2.0
zfp@0.5.5%gcc@11.2.0

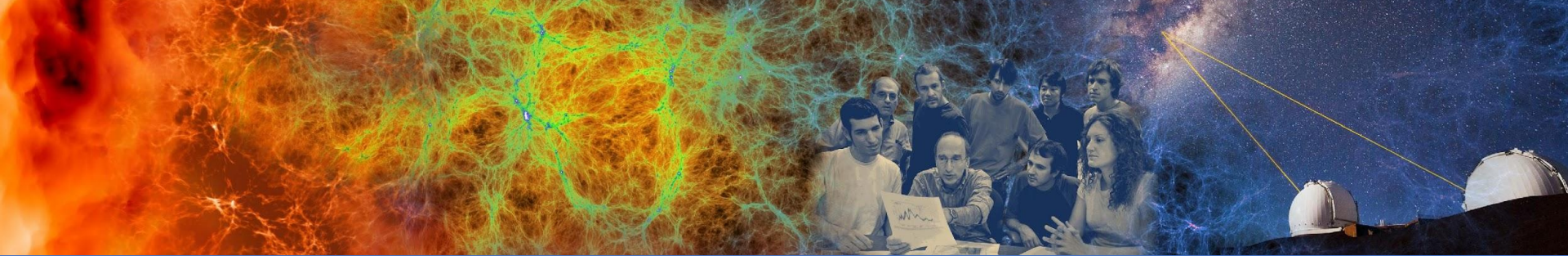
==> Installed packages
-- linux-sles15-zen3 / gcc@11.2.0 -----
gsl@2.7.1-external-cblas build.system=autotools
==> 1 installed package
epalmer># There it is
epalmer># To get the location of the libraries I need I will use a spack command
and store the output in a bash variable
epalmer>export GSL_ROOT=$(spack location -i gsl@2.7.1)
epalmer>echo $GSL_ROOT
/global/common/software/spackecp/perlmutter/e4s-22.11/03104/spack/opt/spack/linu
x-sles15-zen3/gcc-11.2.0/gsl-2.7.1-usionb0cmq56cvupqse43j4yv275154d
epalmer>ls $GSL_ROOT
bin include lib share
epalmer># So I can use the variable to access the location
epalmer># Now I will compile
epalmer>
```

Suppose my example code, `gsl_test.cpp`, requires `gsl` ver. 2.7.1.



Programming Environments and Compilation Best Practices:

- Use `module spider`
- Use compiler wrappers `–CC`, `cc`, and `ftn`– with PrgEnv modules
- Verify CMake builds with `ccmake`
- man files are the source of truth, i.e. `man intro_mpi`
- See examples builds at <https://github.com/NERSC/community-software>
- More questions? Need help? ... <http://help.nersc.gov/>



Thanks for your Attention!

Documentation for Programming Environments and Compilation:

- Modules – <https://docs.nersc.gov/environment/lmod/>
- Programming Environment & Compilers – <https://docs.nersc.gov/systems/perlmutter/#compilingbuilding-software>
- Shell Scripts – https://docs.nersc.gov/environment/shell_startup/
- Spack – <https://docs.nersc.gov/development/build-tools/spack/>
- E4S – <https://docs.nersc.gov/applications/e4s/perlmutter/22.05/>

More questions? Need help? ... <http://help.nersc.gov/>



BERKELEY LAB



U.S. DEPARTMENT OF
ENERGY

Office of
Science