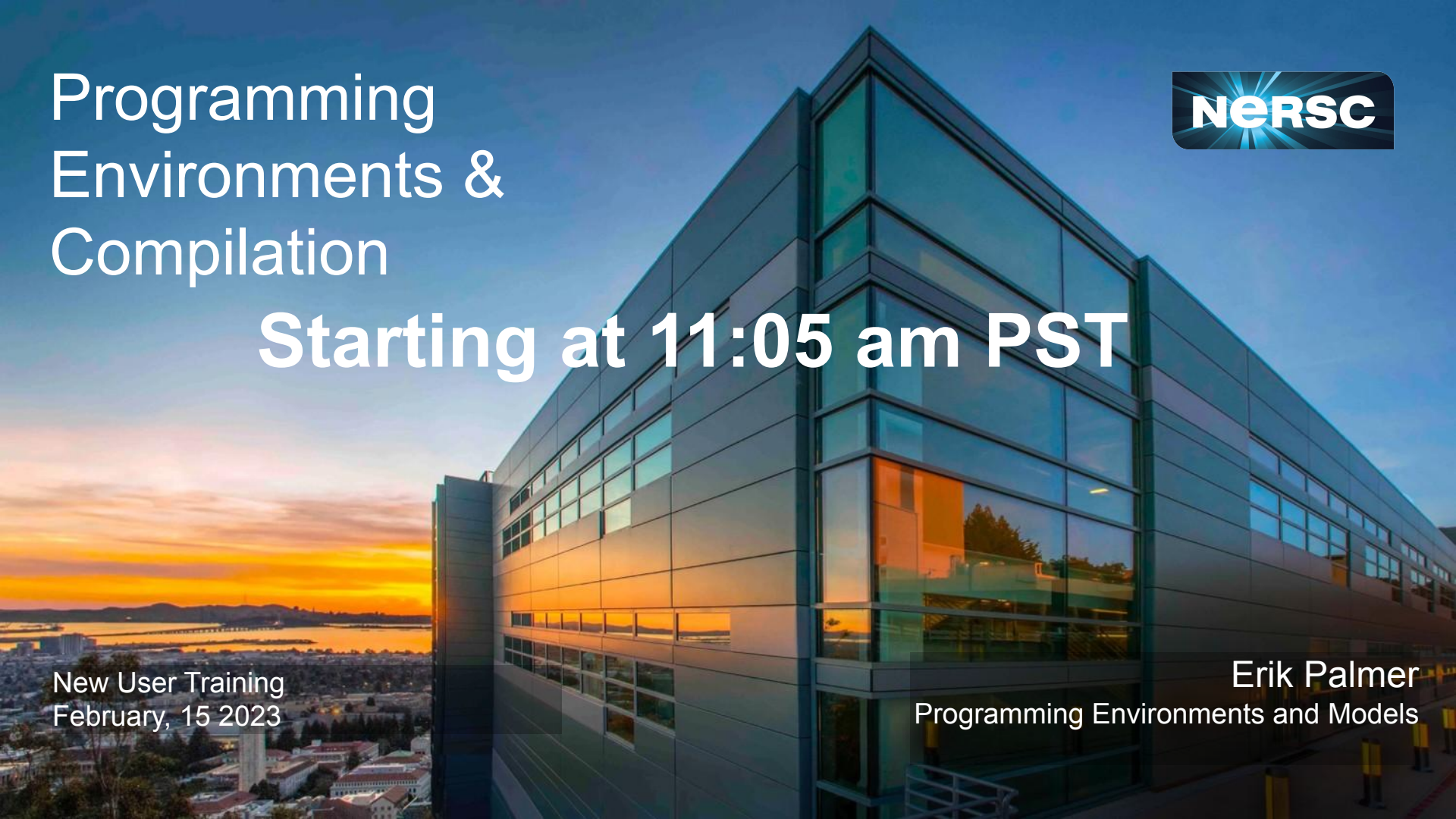
Programming Environments & Compilation



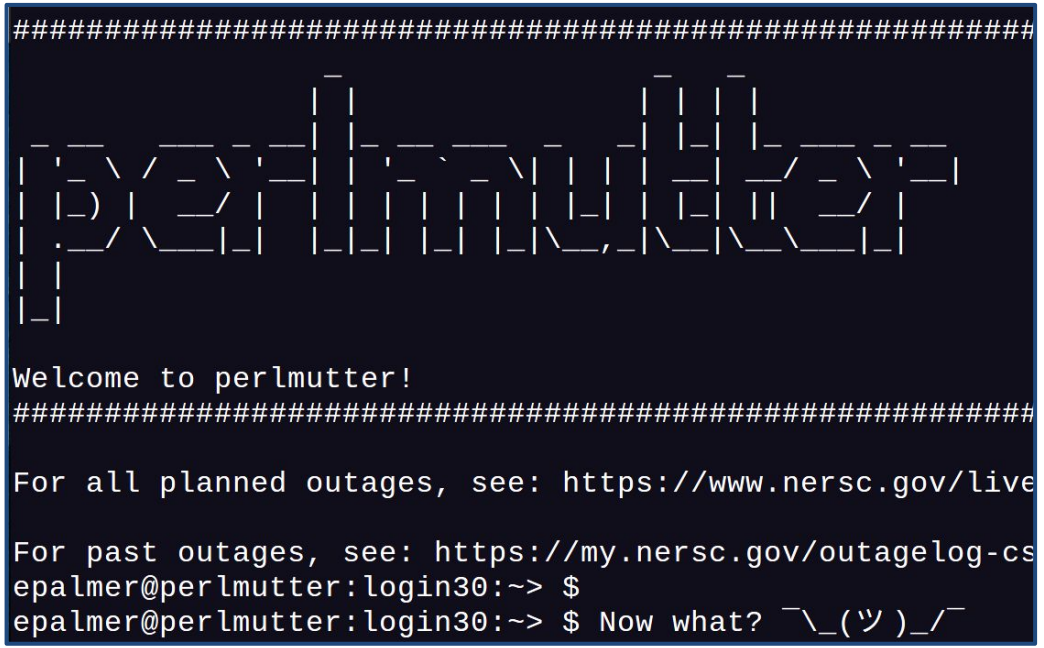
Starting at 11:05 am PST

New User Training
February, 15 2023

Erik Palmer
Programming Environments and Models

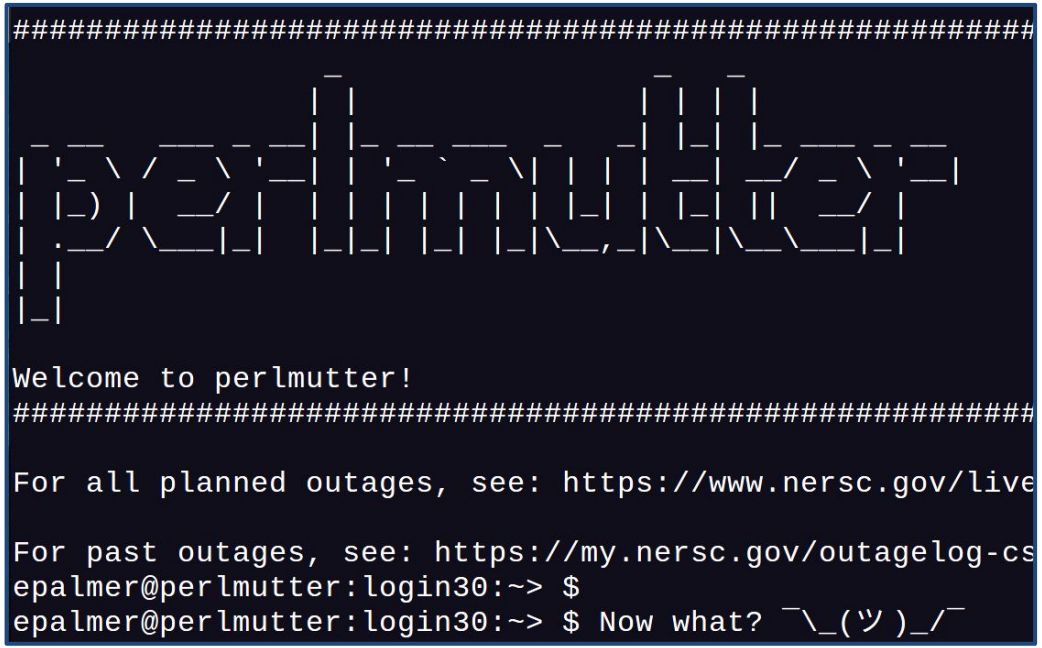
Outline: Four Main Ways of Getting Software

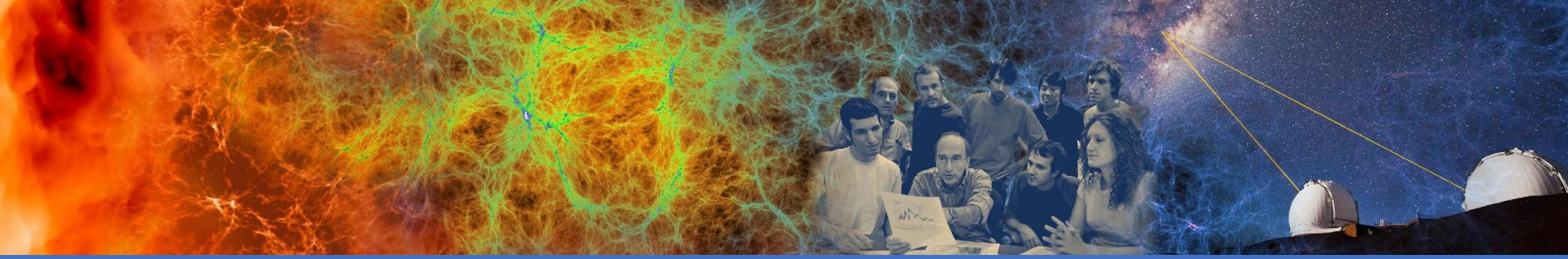
- **Loading modules**
- **Containers**
- **Compiling from source**
- **Conda, Spack & E4S**



Outline: Four Main Ways of Getting Software

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





Modules: Loading Preinstalled Software

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-nvidia:
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-dsmml: cray-dsmml/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-ofed:
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-mrnet:
cray-netcdf:
cray-netcdf-hdf5parallel: cray-netcdf-hdf5parallel/4.9.0.9
cray-openshmemx:
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-nvidia70: craype-accel-nvidia70
craype-accel-nvidia80: craype-accel-nvidia80
craype-accel-nvidia90: craype-accel-nvidia90
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:
gsl: gsl/2.7
hip:
idl:
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: 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:
settarg: settarg
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
2) libfabric/1.15.2.0
3) craype-network-ofi
4) xpmem/2.6.2-2.5_2.38__gd067c3f.shasta
5) PrgEnv-gnu/8.5.0
6) cray-dsmml/0.2.2
7) cray-libsci/23.12.5
8) cray-mpich/8.1.28
9) craype/2.7.30
10) gcc-native/12.3
11) perftools-base/23.12.0
12) cpe/23.12
13) cudatoolkit/12.2
14) craype-accel-nvidia80
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`

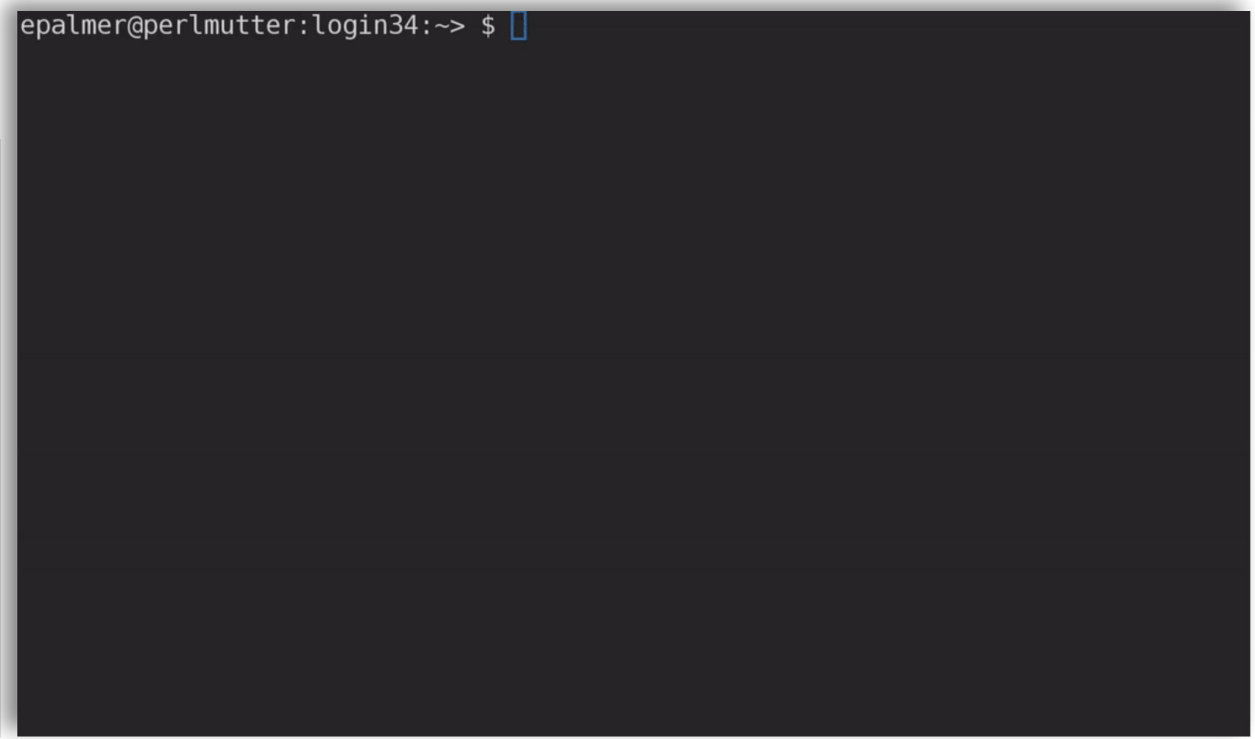
More information: `man module` or https://docs.nersc.gov/environment/lmod/

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**



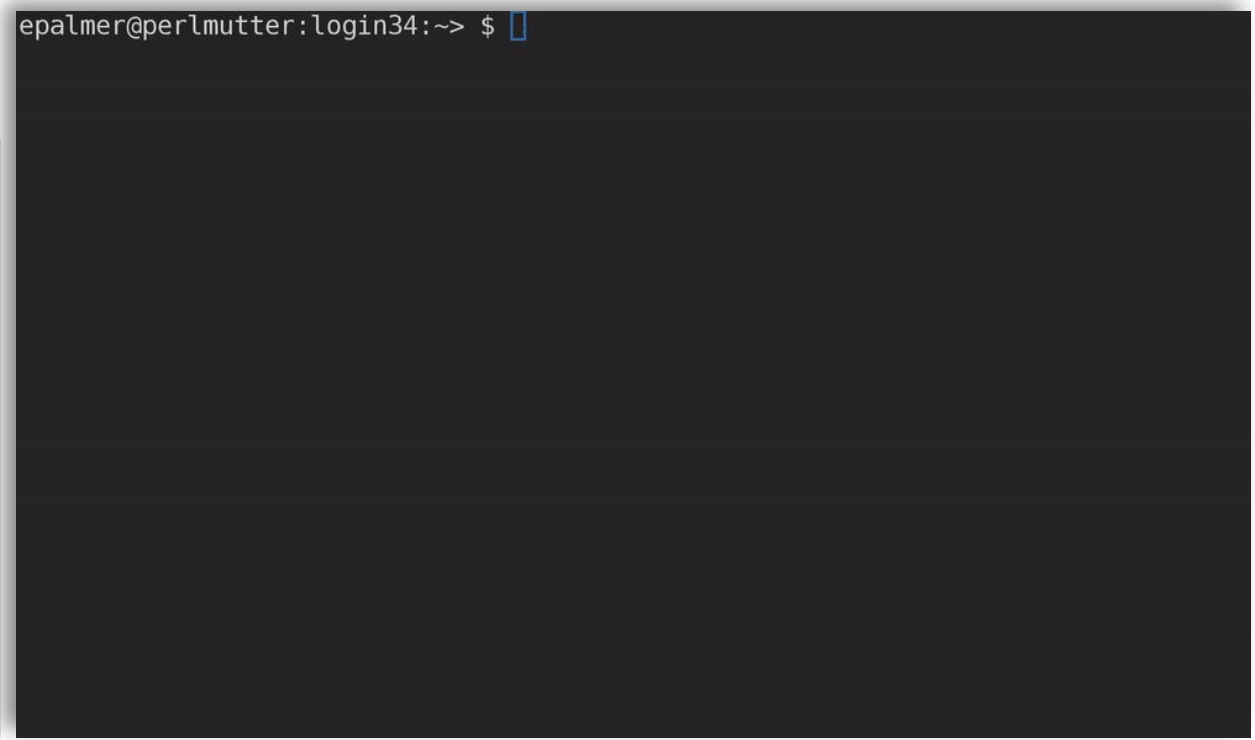
More information: `man module` or https://docs.nersc.gov/environment/lmod/

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**



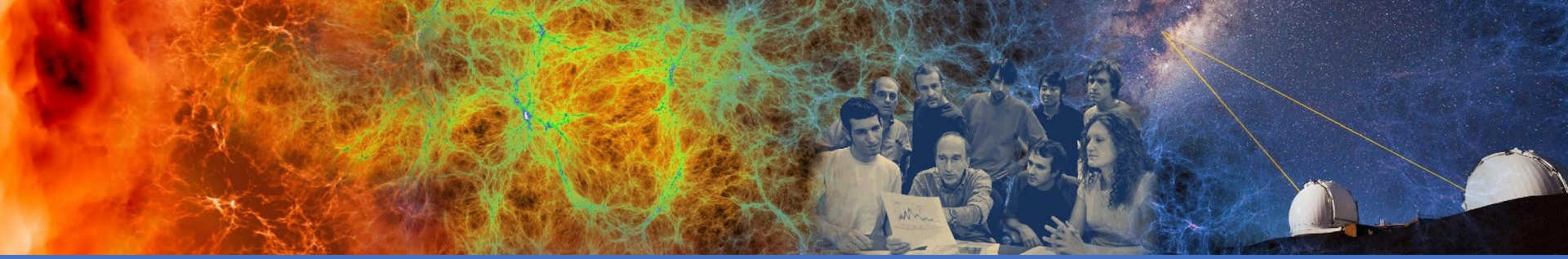
More information: `man module` or https://docs.nersc.gov/environment/lmod/

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 analyzin
prepend_path("PATH","/opt/cray/pe/hdf5/1.12.2.9/bin")
prepend_path("PKG_CONFIG_PATH","/opt/cray/pe/hdf5/1.12.2.9/gnu/12.3/lib/pkgconfig")
prepend_path("PE_PKGCONFIG_LIBS","hdf5_hl:hdf5")
setenv("PE_HDF5_PKGCONFIG_LIBS","hdf5_hl:hdf5")
prepend_path("PE_FORTRAN_PKGCONFIG_LIBS","hdf5hl_fortran:hdf5_fortran")
setenv("PE_HDF5_FORTRAN_PKGCONFIG_LIBS","hdf5hl_fortran:hdf5_fortran")
prepend_path("PE_CXX_PKGCONFIG_LIBS","hdf5_hl_cpp:hdf5_cpp")
setenv("PE_HDF5_CXX_PKGCONFIG_LIBS","hdf5_hl_cpp:hdf5_cpp")
setenv("CRAY_HDF5_DIR","/opt/cray/pe/hdf5/1.12.2.9")
setenv("PE_HDF5_DIR","/opt/cray/pe/hdf5/1.12.2.9")
setenv("CRAY_HDF5_VERSION","1.12.2.9")
setenv("CRAY_HDF5_PREFIX","/opt/cray/pe/hdf5/1.12.2.9/gnu/12.3")
setenv("HDF5_DIR","/opt/cray/pe/hdf5/1.12.2.9/gnu/12.3")
setenv("HDF5_ROOT","/opt/cray/pe/hdf5/1.12.2.9/gnu/12.3")
prepend_path("CRAY_LD_LIBRARY_PATH","/opt/cray/pe/hdf5/1.12.2.9/gnu/12.3/lib")
prepend_path("MODULEPATH","/opt/cray/pe/lmod/modulefiles/hdf5/gnu/12.0/cray-hdf5/1.12.2")
```

Path Changes
Environment Variables
Other Info

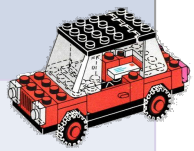


Programming Environments: Configuring Compilers and Libraries

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/ofi/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/ofi/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/ofi/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,-lpthread,--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
-I/opt/cray/pe/mpich/8.1.28/ofi/nvidia/23.3/include …
```

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

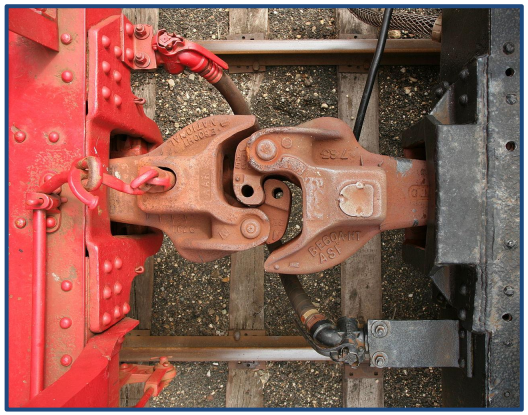
```
icpx -march=core-avx2 -mtune=core-avx2 -D__CRAY_X86_MILAN -D__CRAY_NVIDIA80
-D__CRAYXT_COMPUTE_LINUX_TARGET helloworld.cpp -o helloworld.ex
-Wl,-rpath=/opt/intel/oneapi/compiler/2023.2.0/linux/compiler/lib/intel64
-I/opt/cray/pe/mpich/8.1.28/ofi/intel/2022.1/include …
```

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

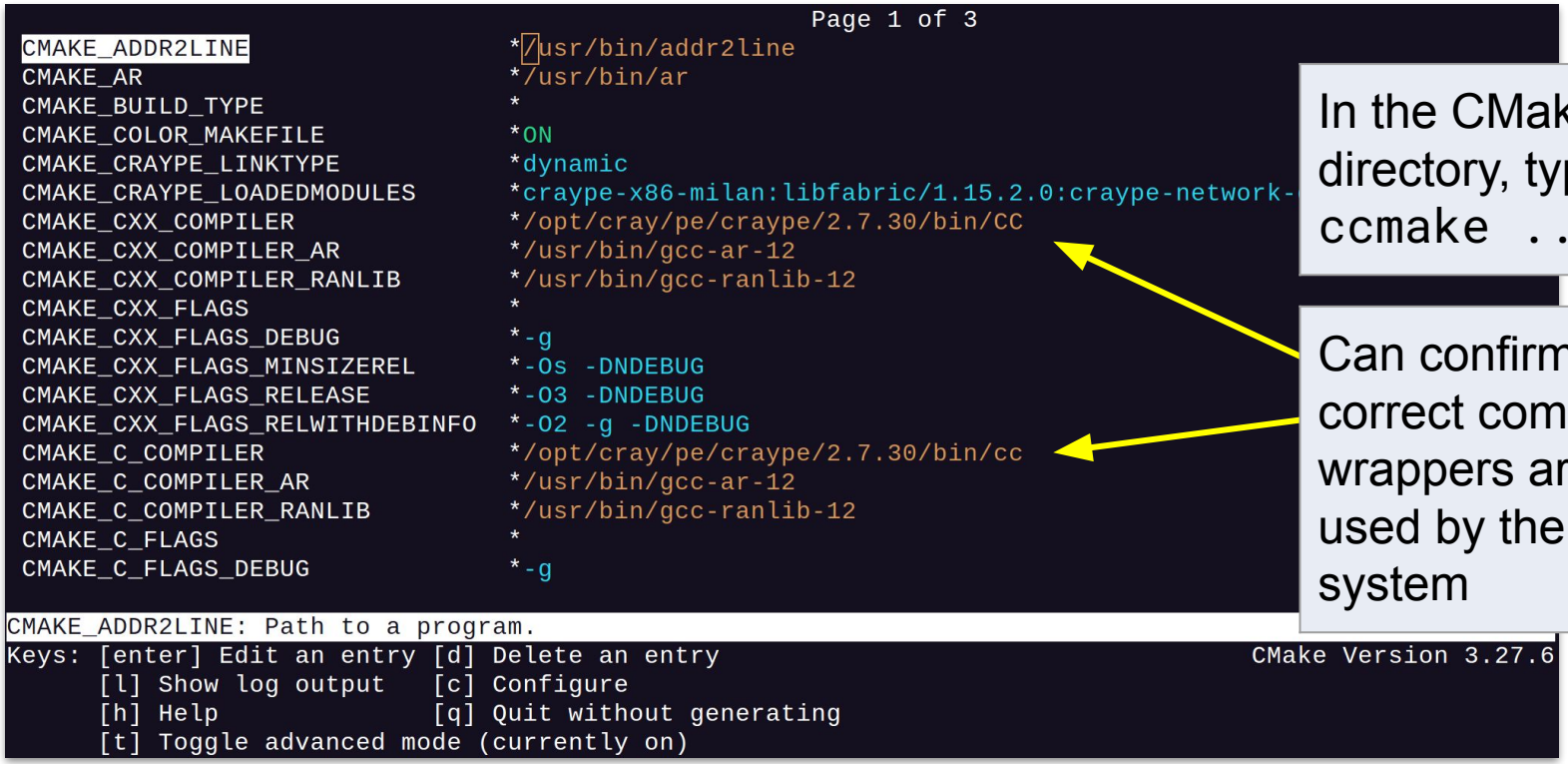
From INSTALL.md:

```
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



In the CMake build directory, type:
`ccmake ..`

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

Two Comments about Linking:

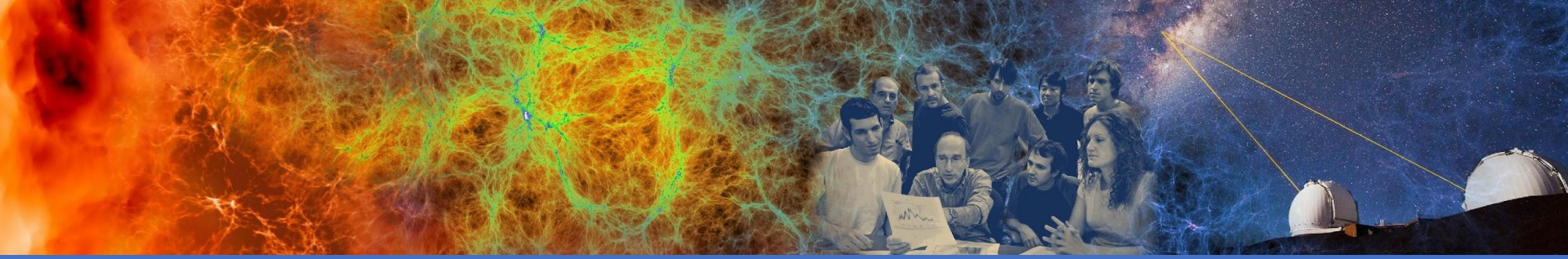
- Many modules prepend the `LIBRARY_PATH` so it is not necessary to specify library locations, e.g.

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

Dynamic vs. Static Linking

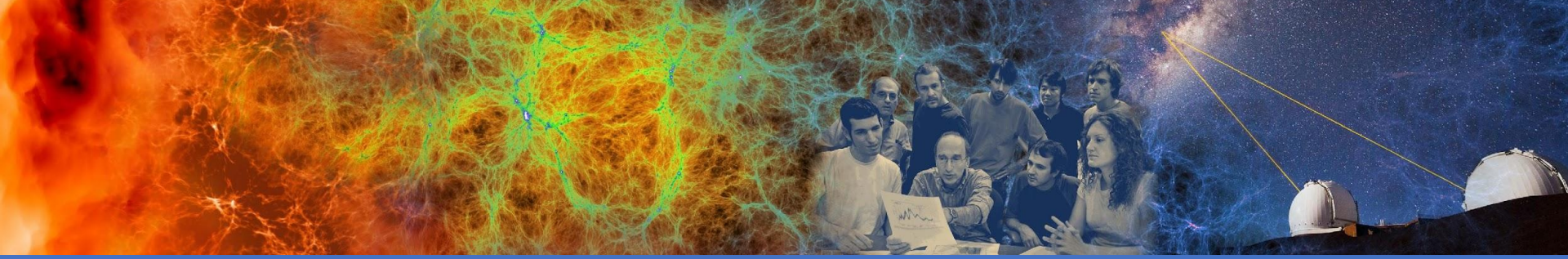
- Cray wrappers build dynamically linked executables by default.
  (The kind that give, "`error while loading shared libraries: ...`" when they get misplaced at runtime.)
- On Perlmutter static compilation with `-static` or `CRAYPE_LINK_TYPE=static` can fail and is not supported.

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/running-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)

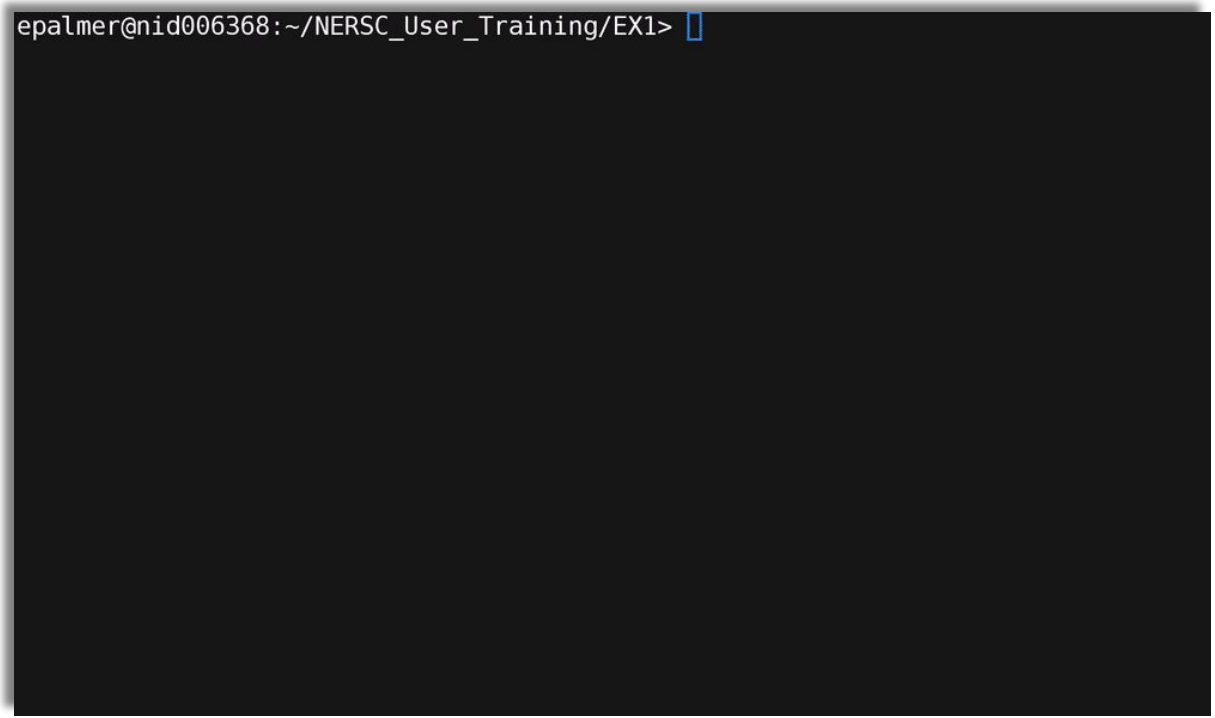
```
epalmer@nid006368:~/NERSC_User_Training/EX1> #In this example, we will compile a
 code with MPI and OpenMP
```

Modules Loaded:
PrgEnv-gnu

Compile line:

```
cc hellohybrid.c -fopenmp -o hellohybrid
```

Compile with MPI and OpenMP (CPU only)



Modules Loaded:
PrgEnv-gnu

Compile line:

```
cc hellohybrid.c -fopenmp -o hellohybrid
```

Compile with MPI and OpenMP on GPU

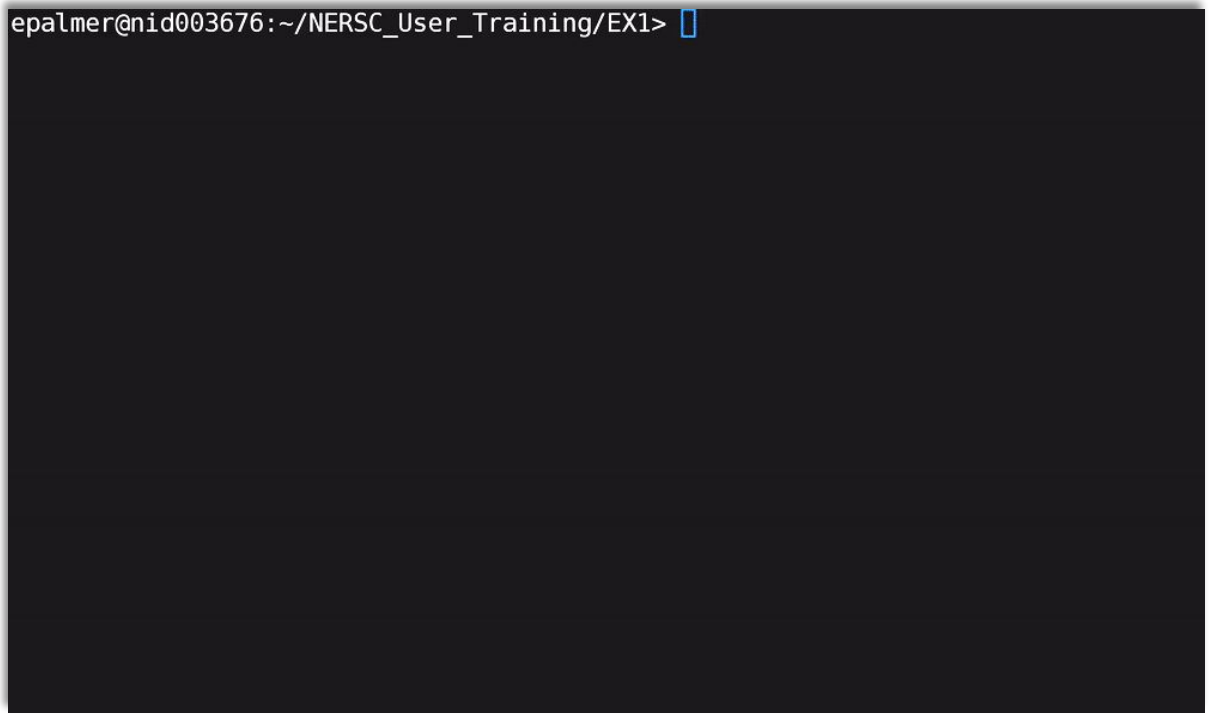
```
epalmer@nid003676:~/NERSC_User_Training/EX1> ls
hellohybrid.c
epalmer@nid003676:~/NERSC_User_Training/EX1> module list

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



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

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.

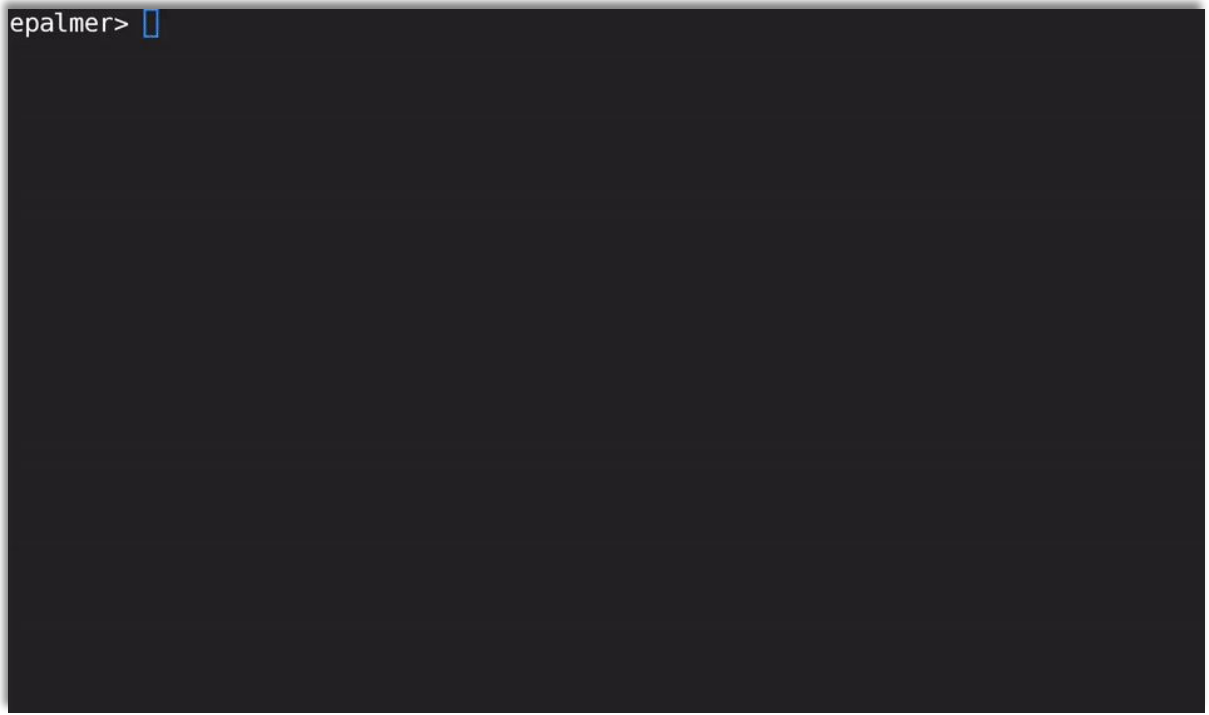
Compile a CUDA Code with CUDA-Aware MPI

```
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



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:

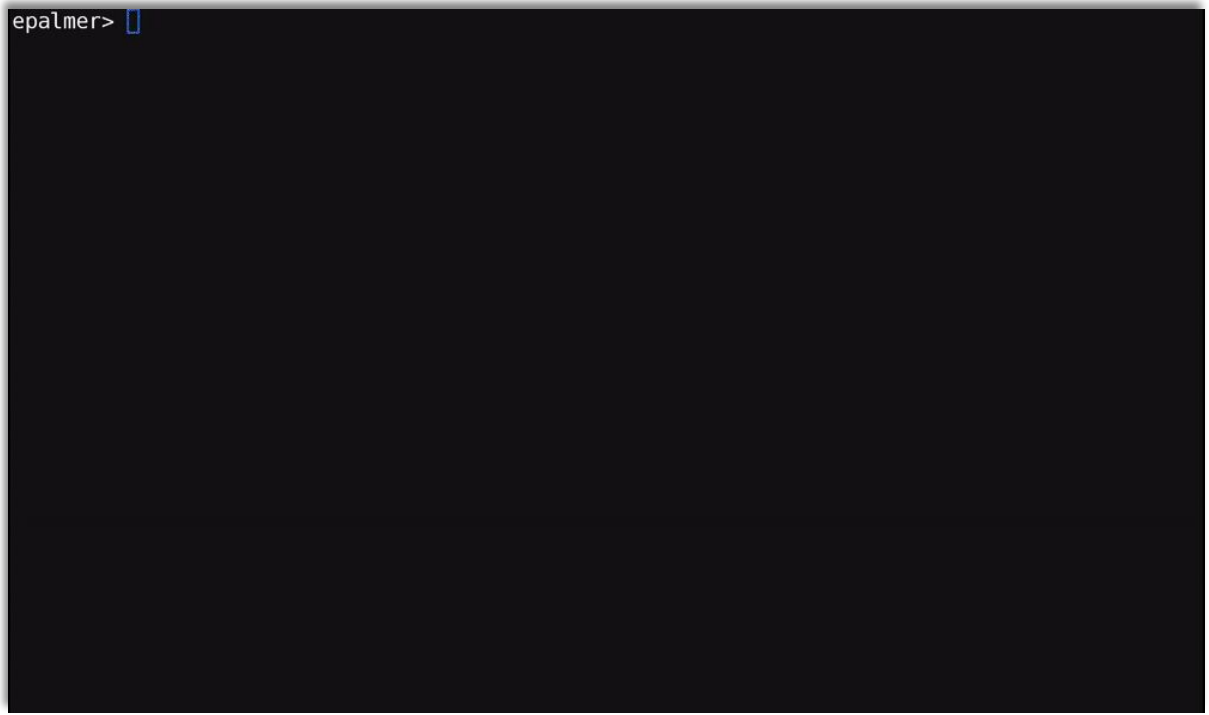
```
cc helloacc.c -acc -Minfo=acc -o helloacc
```

*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



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

OpenACC
More Science, Less Programming

MPICH

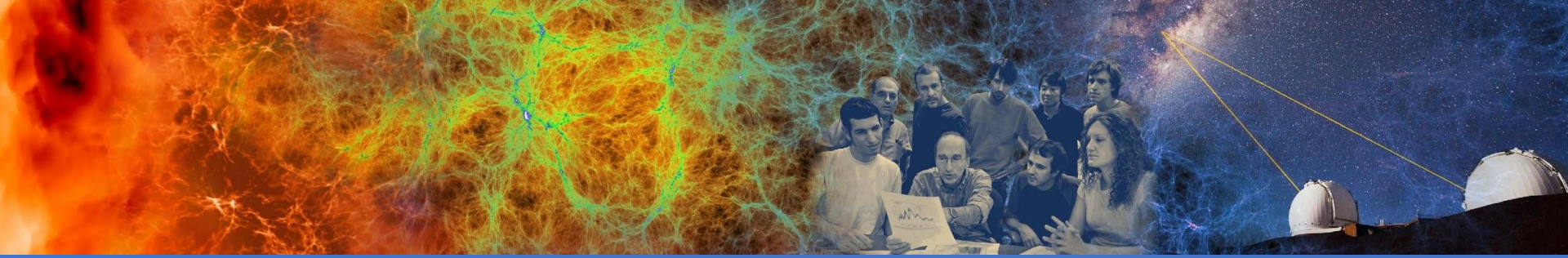
Stdpar

Stdpar

HIP

OpenMP

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



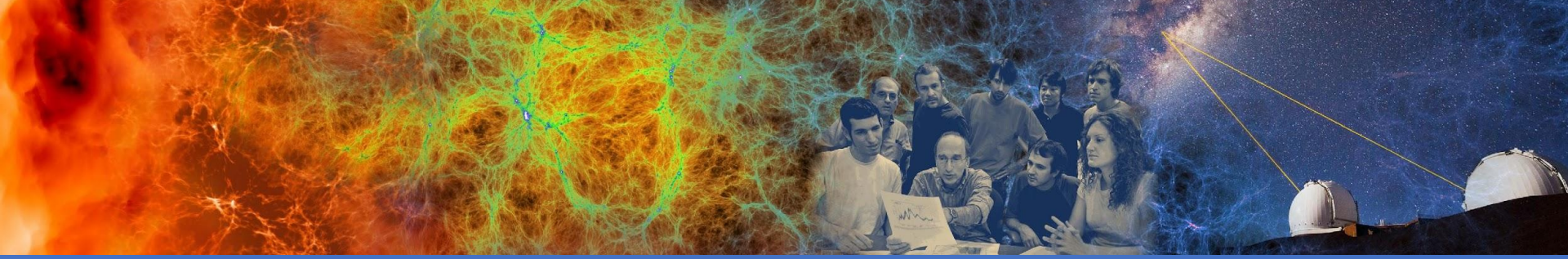
Where to install your code

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

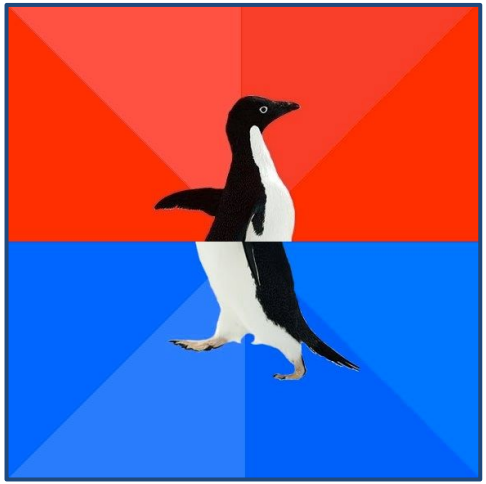
NERSC

BERKELEY LAB

U.S. DEPARTMENT OF ENERGY | Office of Science

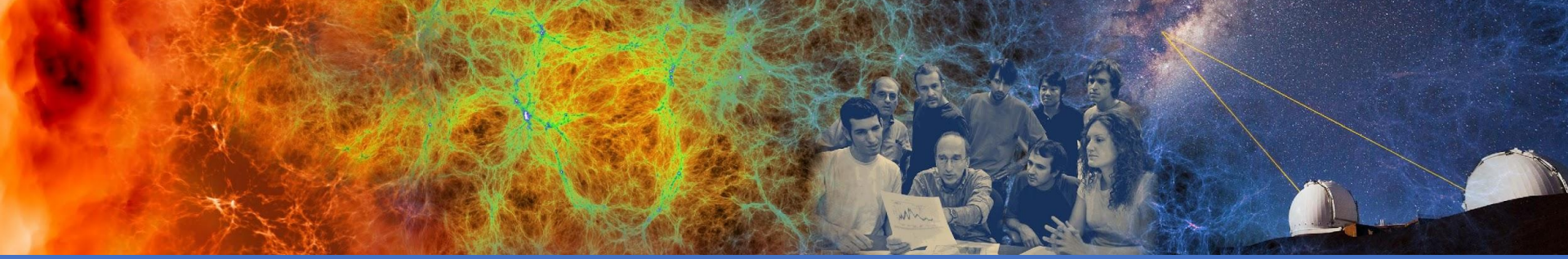
Installing a package with Spack

```
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!

NERSC

BERKELEY LAB

U.S. DEPARTMENT OF ENERGY | Office of Science

The Extreme-Scale Scientific Software Stack (E4S)

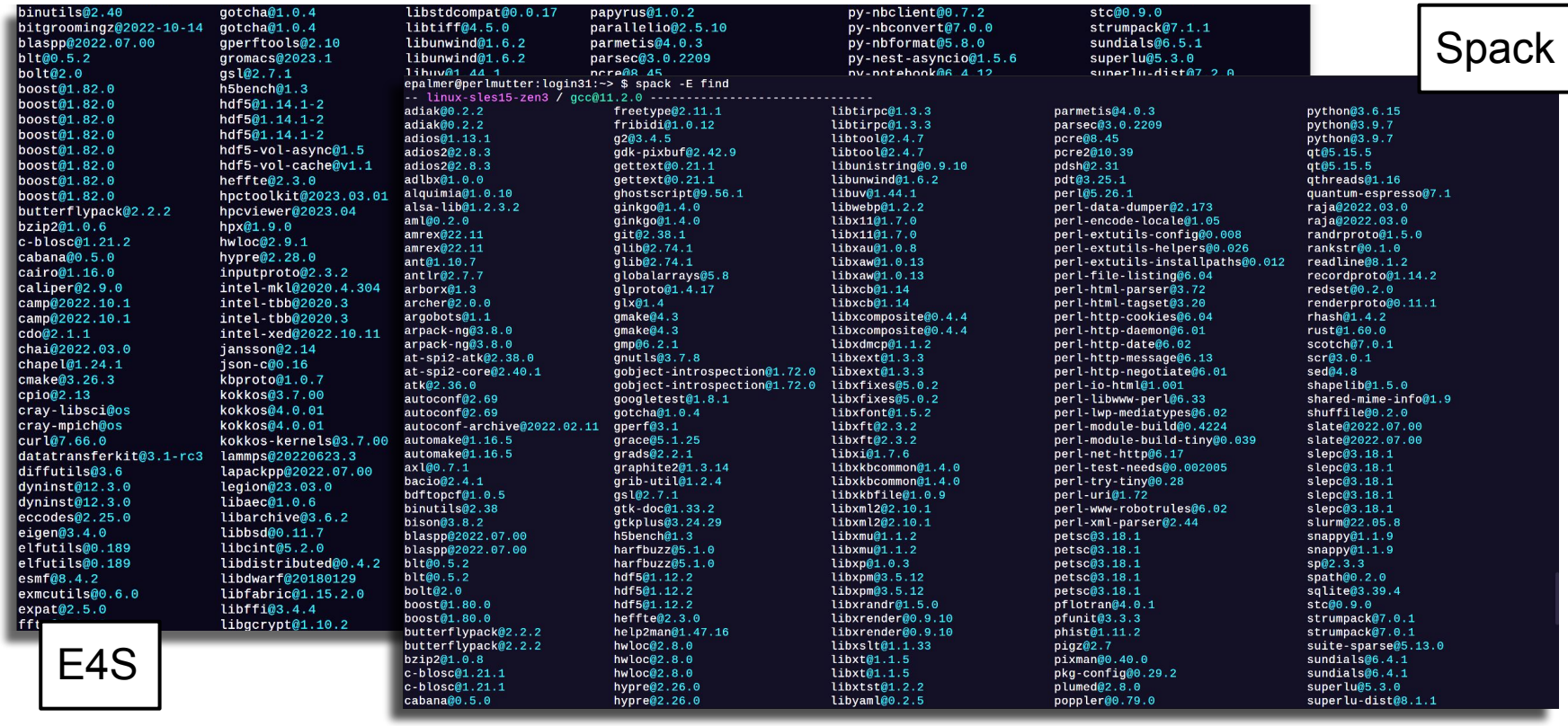
module	Version	Command	Installed Packages	Can I Install More?
e4s	23.08	module load e4s	**478**	Yes

```
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

E4S

NERSC

BERKELEY LAB
Bringing Science Solutions to the World

U.S. DEPARTMENT OF ENERGY | Office of Science

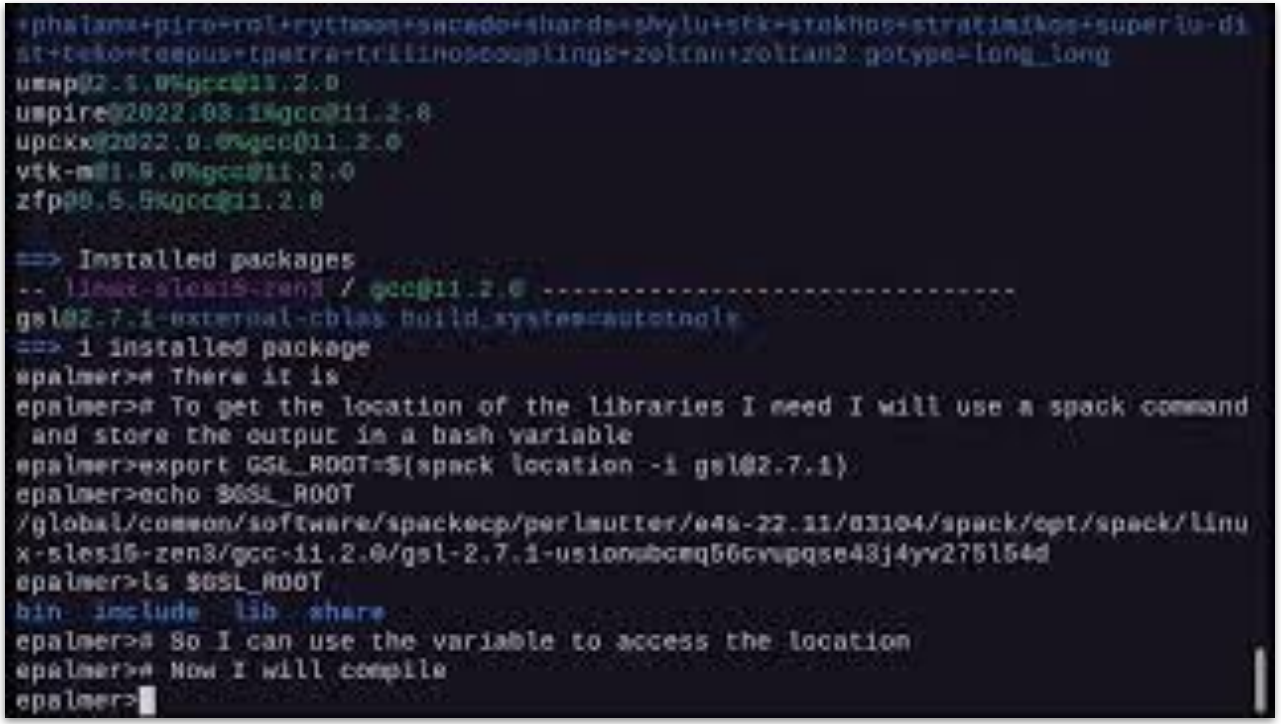
Install or Load a Spack Package

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

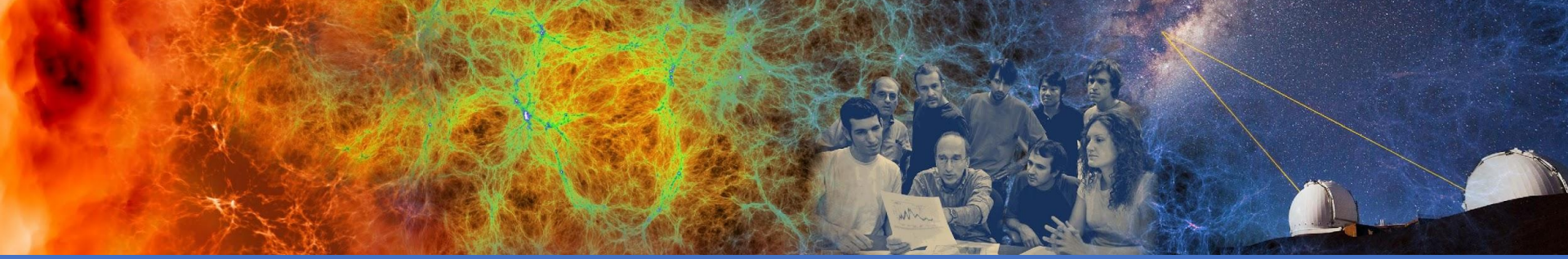
Bonus: `spack load --sh <package>`

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

Example: Install a Dependency with Spack

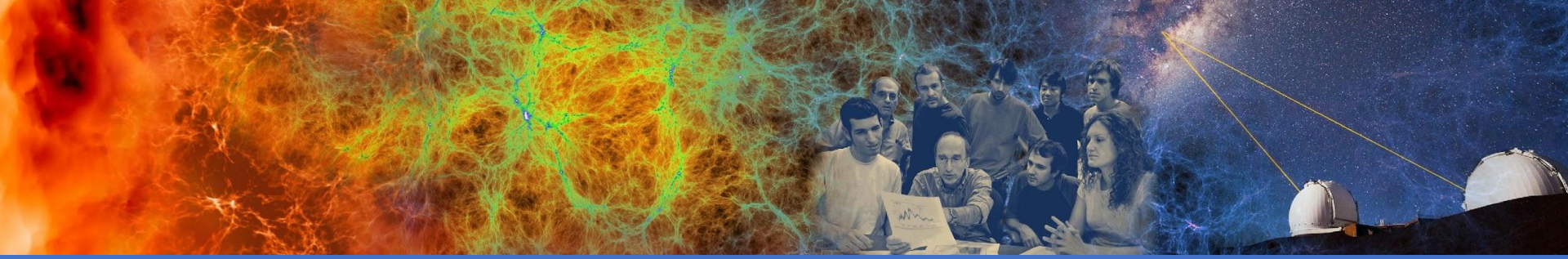


Suppose my example code, `gsl_test.cpp`, requires gsl ver. 2.7.1.



Programing 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/**