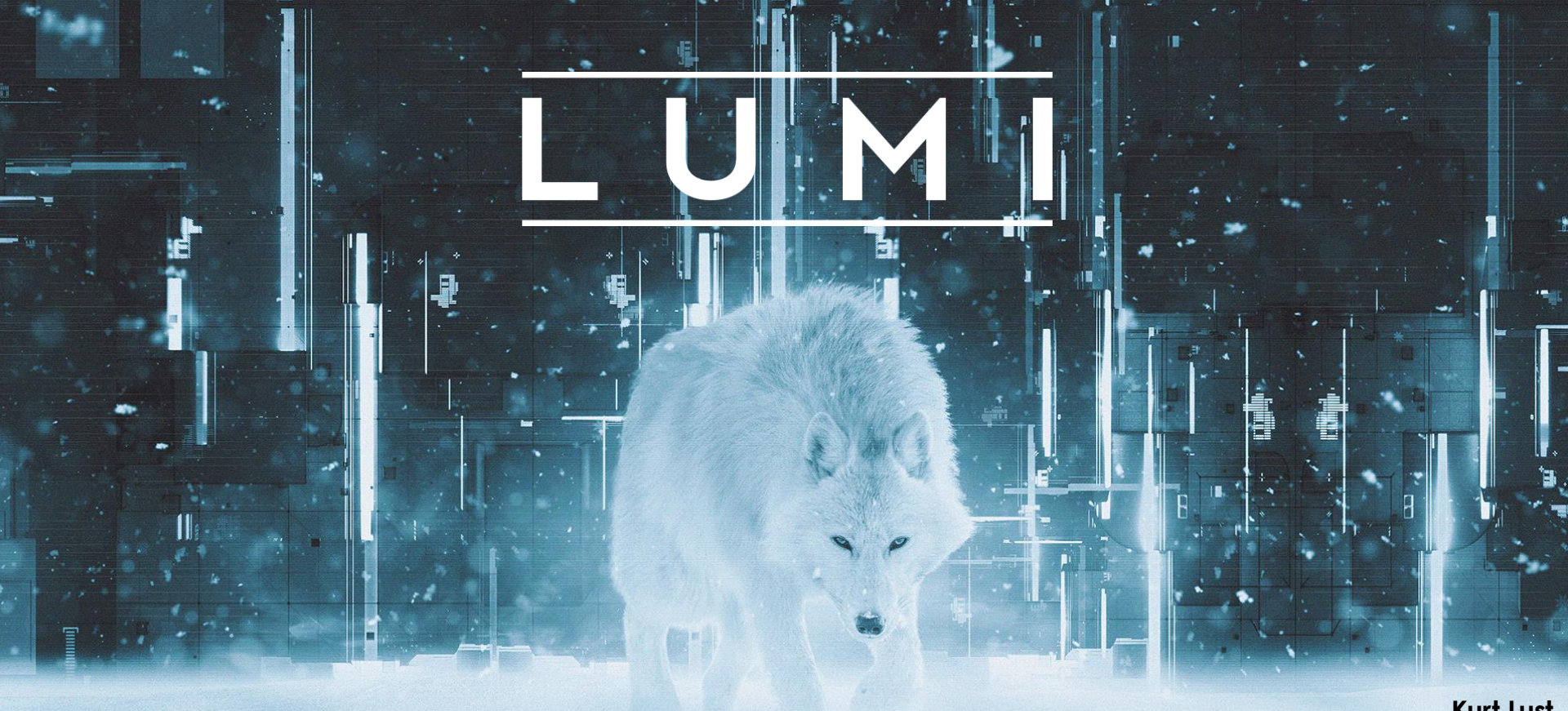


Environment modules on LUMI

Kurt Lust
LUMI User Support Team (LUST)
University of Antwerp

April 2026

Environment modules implementations

LUMI

- Environment modules are used on HPC systems to enable users to create custom environments and select between multiple versions of applications
  - Applications on HPC systems are installed in non-standard places
  - Can set environment variables for the application also
- 3 systems in use
  - Original module tool written in C with modules in Tcl, development halted
  - New implementation in Tcl with many new features, developed at CÉA
    - Not supported by HPE Cray
  - Lmod, an implementation in Lua with native module files in Lua but support for most Tcl module files
- We chose Lmod for LUMI

Exploring modules with Lmod

LUMI

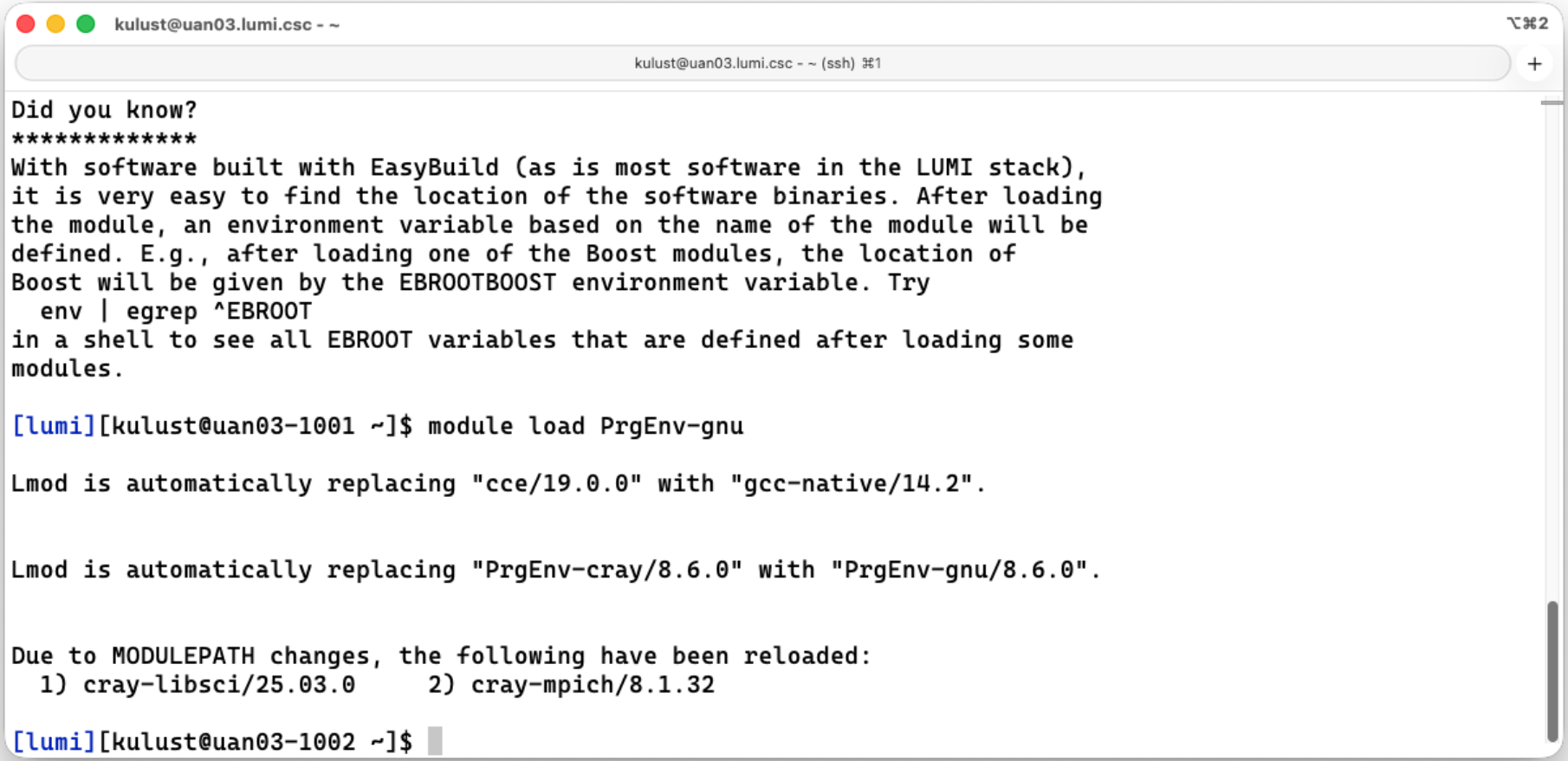
- Contrary to some other module systems, not all modules are immediately available for loading
  - Installed modules: All modules on the system that can be loaded one way or another
  - Available modules: Can be loaded without first loading another module
- Examples in the HPE Cray PE:
  - `cray-mpich` requires a compiler module and network target module first
  - Many of the performance monitoring tools require `perftools-base` first
  - `cray-fftw` only becomes available when a processor target module is loaded
- Tools
  - `module avail` searches in the available modules
  - `module spider` and `module keyword` search in the installed modules
    - But with some restrictions on LUMI

Benefits of a hierarchy

LUMI

- When well designed, you get some protection from loading modules that do not work together well
  - Only partially implemented on LUMI
- When "swapping" a module that makes other modules available with a different one, Lmod will try to look for equivalent modules in the new hierarchy
  - Example: Try `module load PrgEnv-gnu` in the default login environment and see what happens

module load PrgEnv-gnu



Module names and families

LUMI

- In Lmod you cannot have two modules with the same name loaded together
  - On LUMI, when loading a new module the other one with the same name will be automatically unloaded
  - Automatic protection from conflicts
- Extension: family concept: No two modules of the same family can be loaded together
  - E.g., make compilers member of the family "compiler"
  - On LUMI, the conflicting module of the same family will be unloaded automatically

Extensions

LUMI

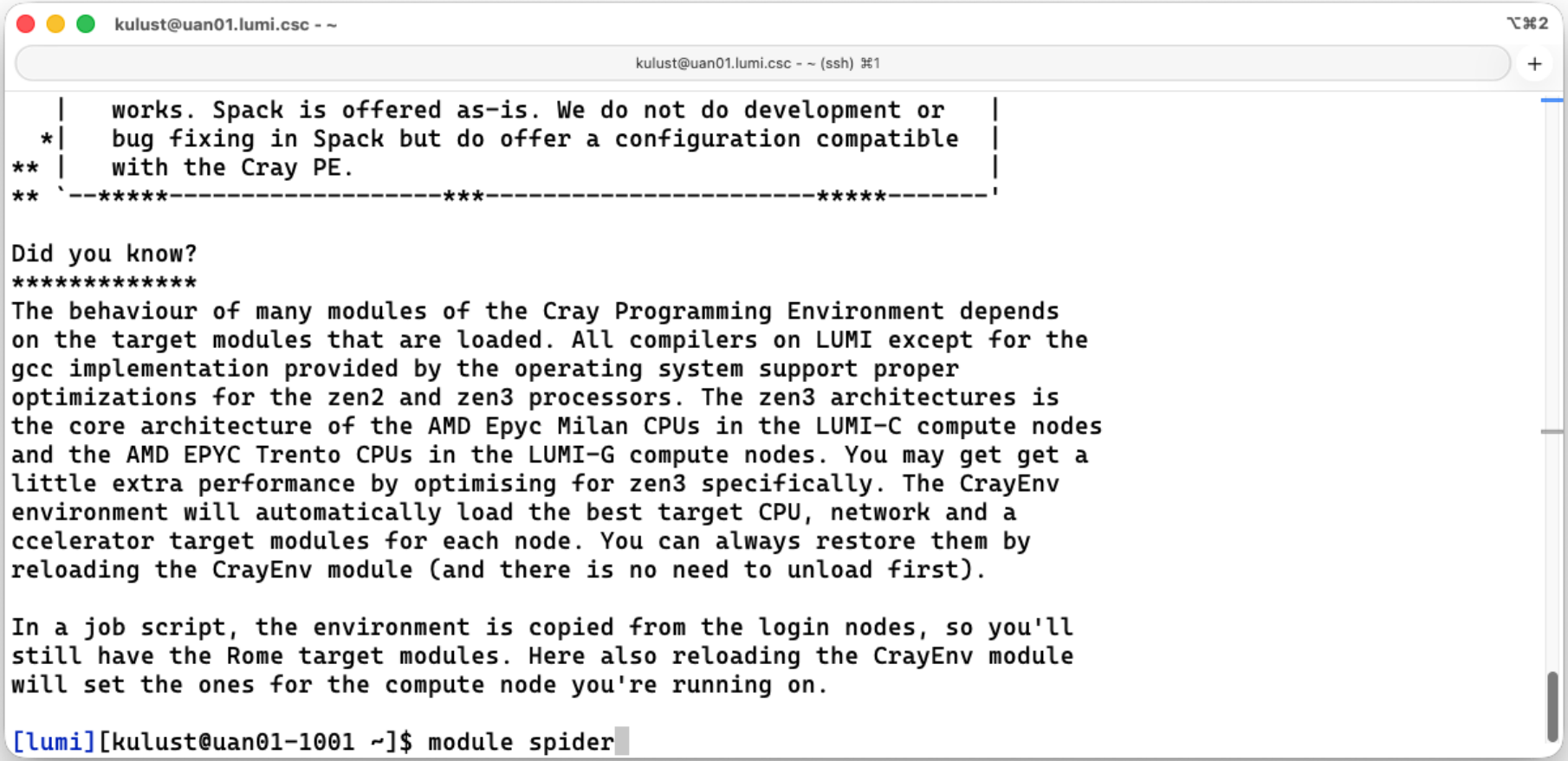
- It would not make sense to have a separate module for each of the hundreds of R packages or tens of Python packages a software stack may contain.
  - Would actually also create a performance problem due to excess metadata access and long PATH variables
  - Bundle related packages in a single module
- Lmod solution: A module can define a list of extensions, basically other packages provided by the module.
  - And the regular commands can be used to search for these
  - Unfortunately not used in the HPE Cray PE cray-python and cray-R modules

module spider

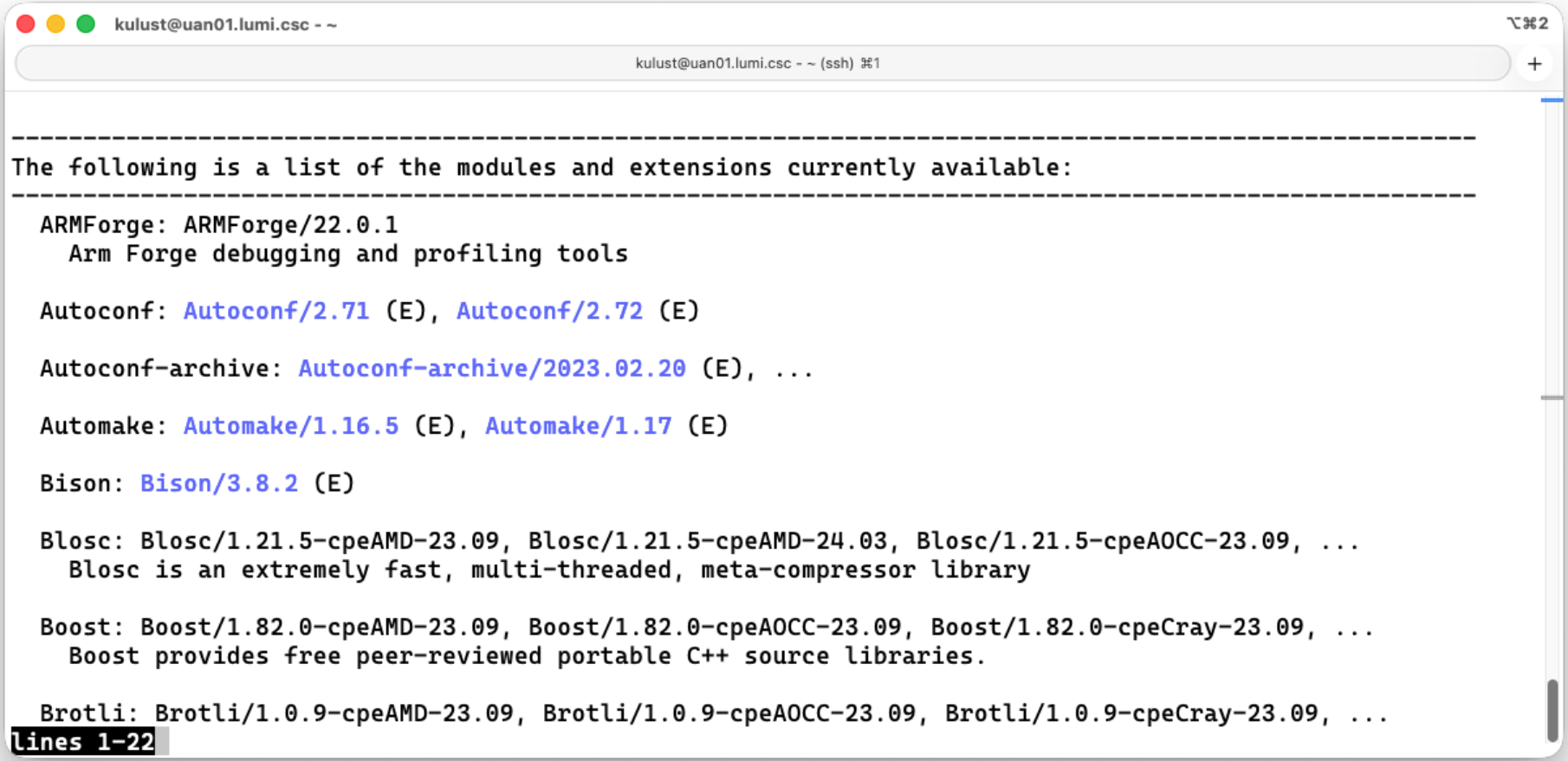
LUMI

- `module spider` : Long list of all installed software with short description
  - Will also look into modules for "extensions" and show those also, marked with an "E"
  - By default only in the main software stacks on LUMI

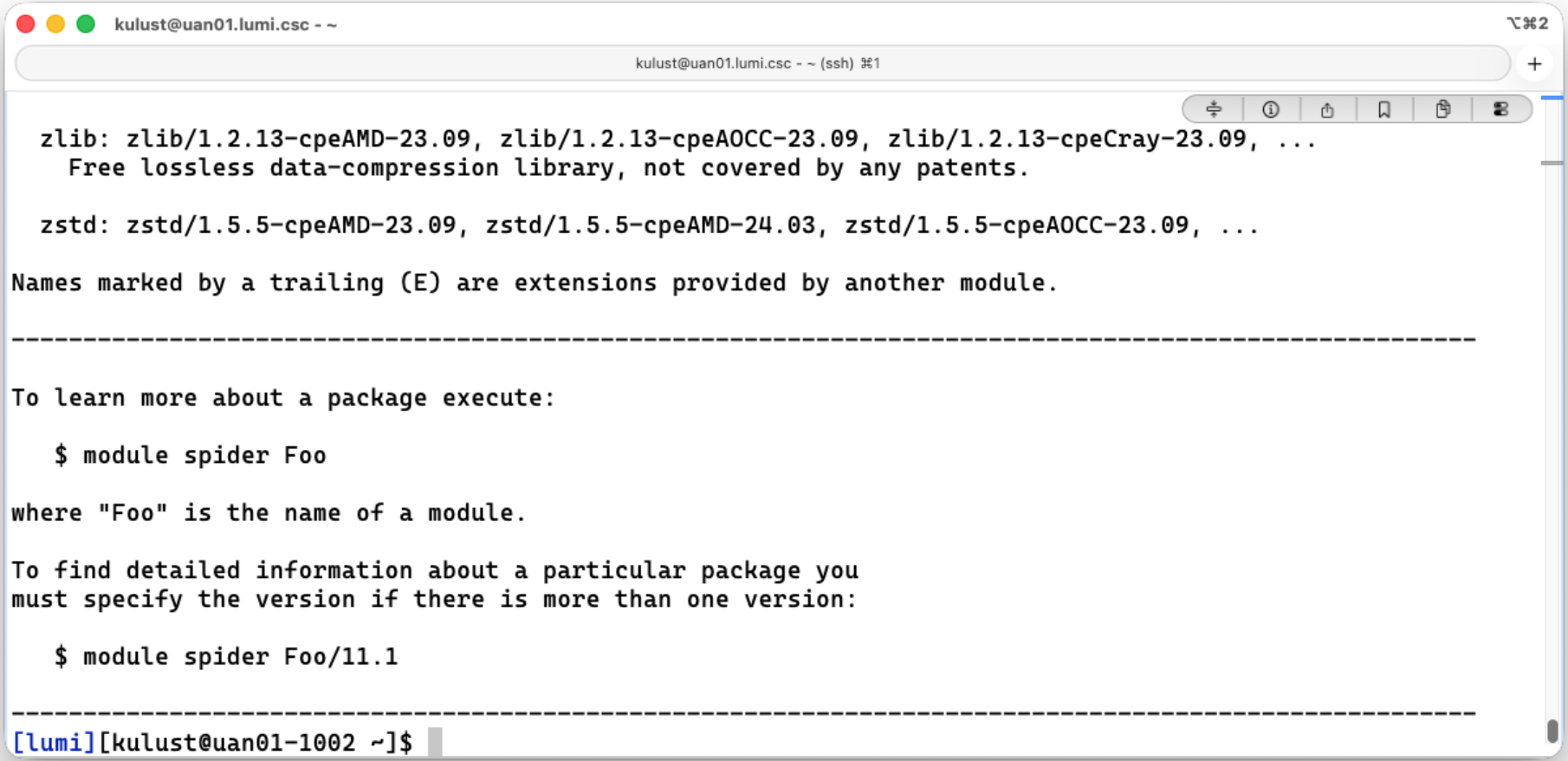
module spider (command)(1)



module spider (command) (2)



module spider (command) (3)

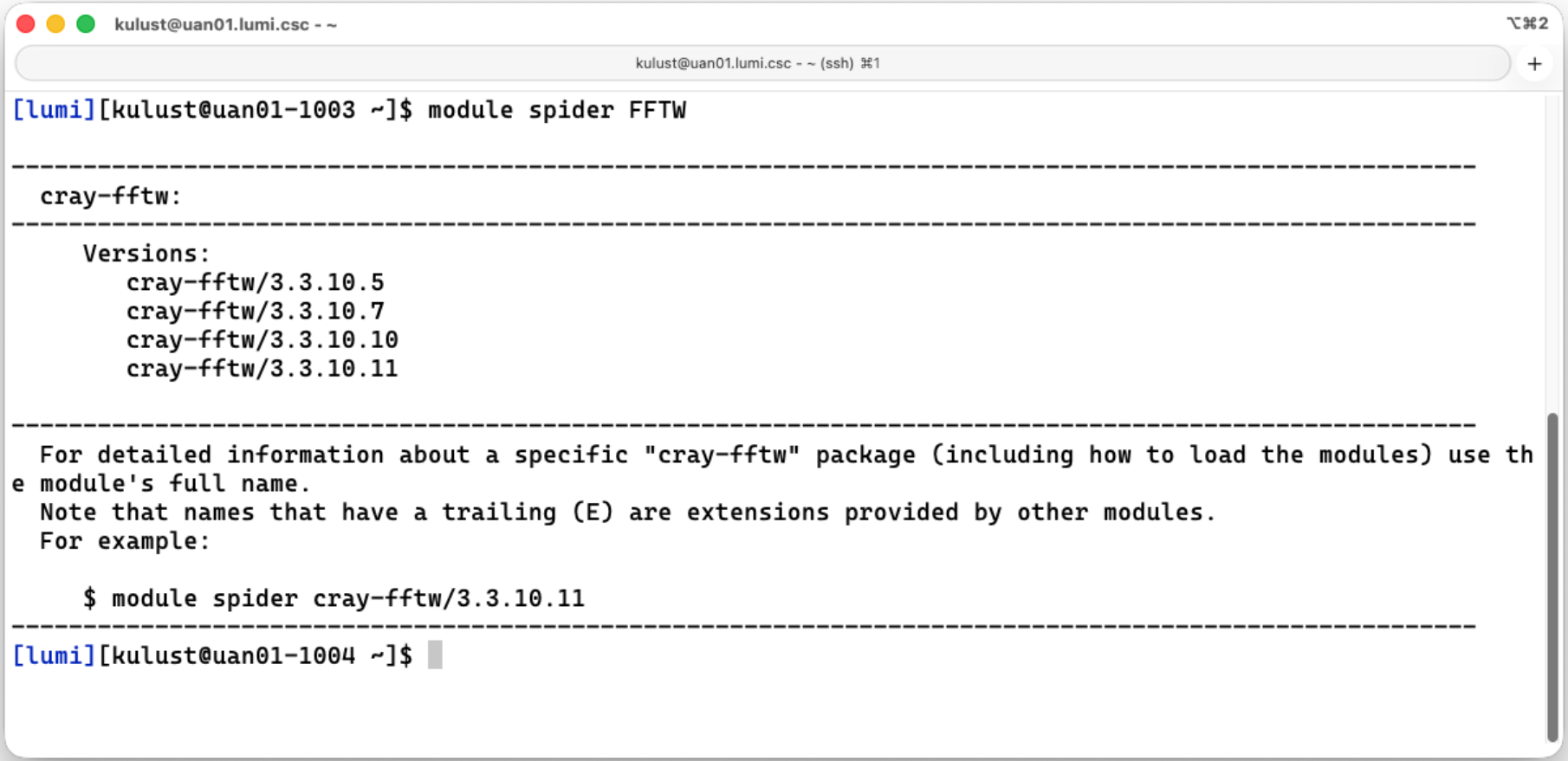


module spider

LUMI

- `module spider` : Long list of all installed software with short description
  - Will also look into modules for "extensions" and show those also, marked with an "E"
  - By default only in the main software stacks on LUMI
- `module spider FFTW` : Look for the FFTW libraries on the system

module spider FFTW

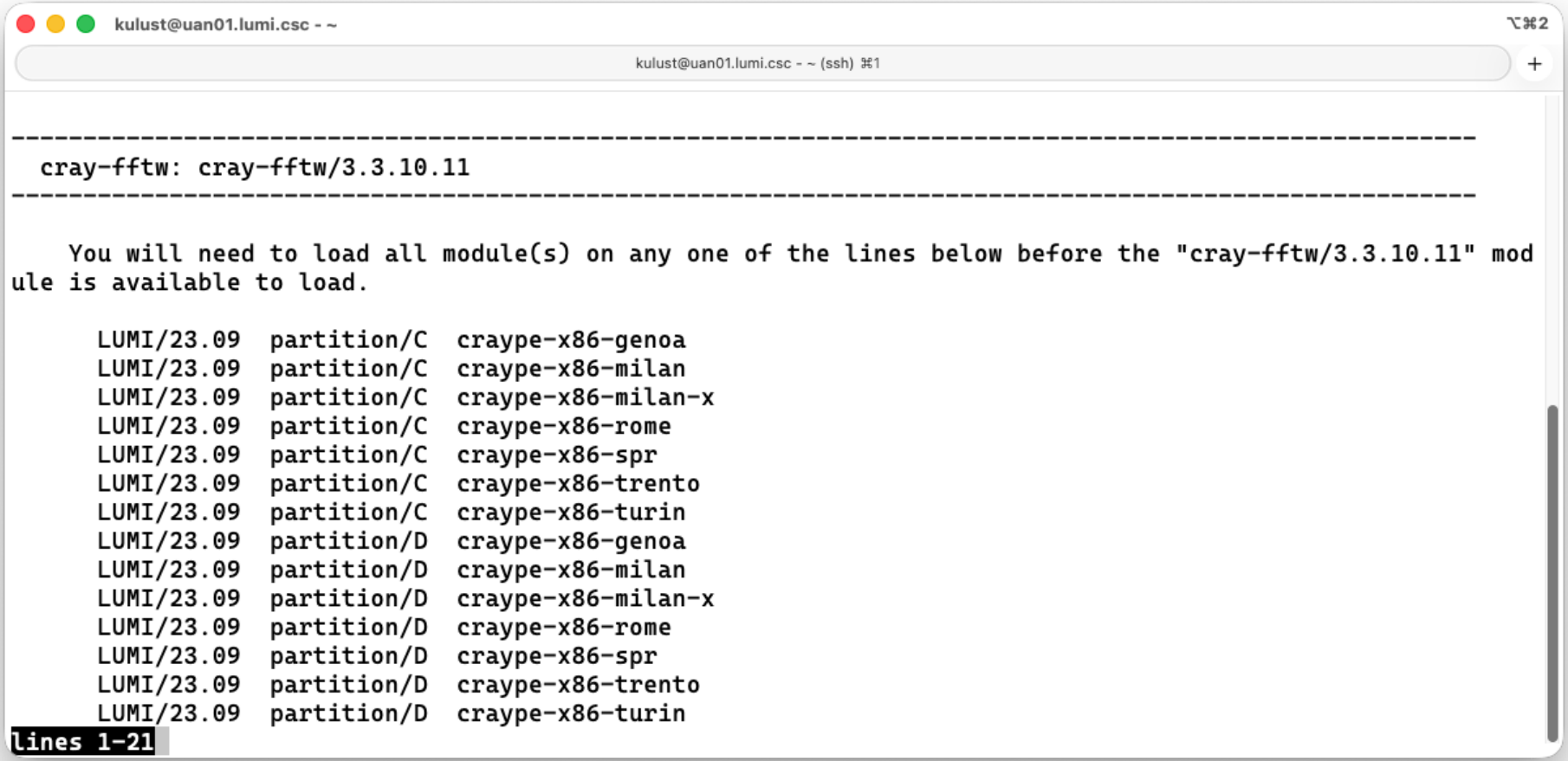


module spider

LUMI

- `module spider` : Long list of all installed software with short description
  - Will also look into modules for "extensions" and show those also, marked with an "E"
  - By default only in the main software stacks on LUMI
- `module spider FFTW` : Look for the FFTW libraries on the system
- `module spider cray-fftw/3.3.10.11`: Look for this specific version
  - But this immediately shows the problems with the HPE Cray PE
    - Some of the lines don't make much sense (see later)
    - Some options are missing also

module spider cray-fftw/3.3.10.11



module spider cray-fftw/3.3.10.11 (2)

LUMI

kulust@uan01.lumi.csc - ~ ⌥⌘2

kulust@uan01.lumi.csc - ~ (ssh) ⌘1

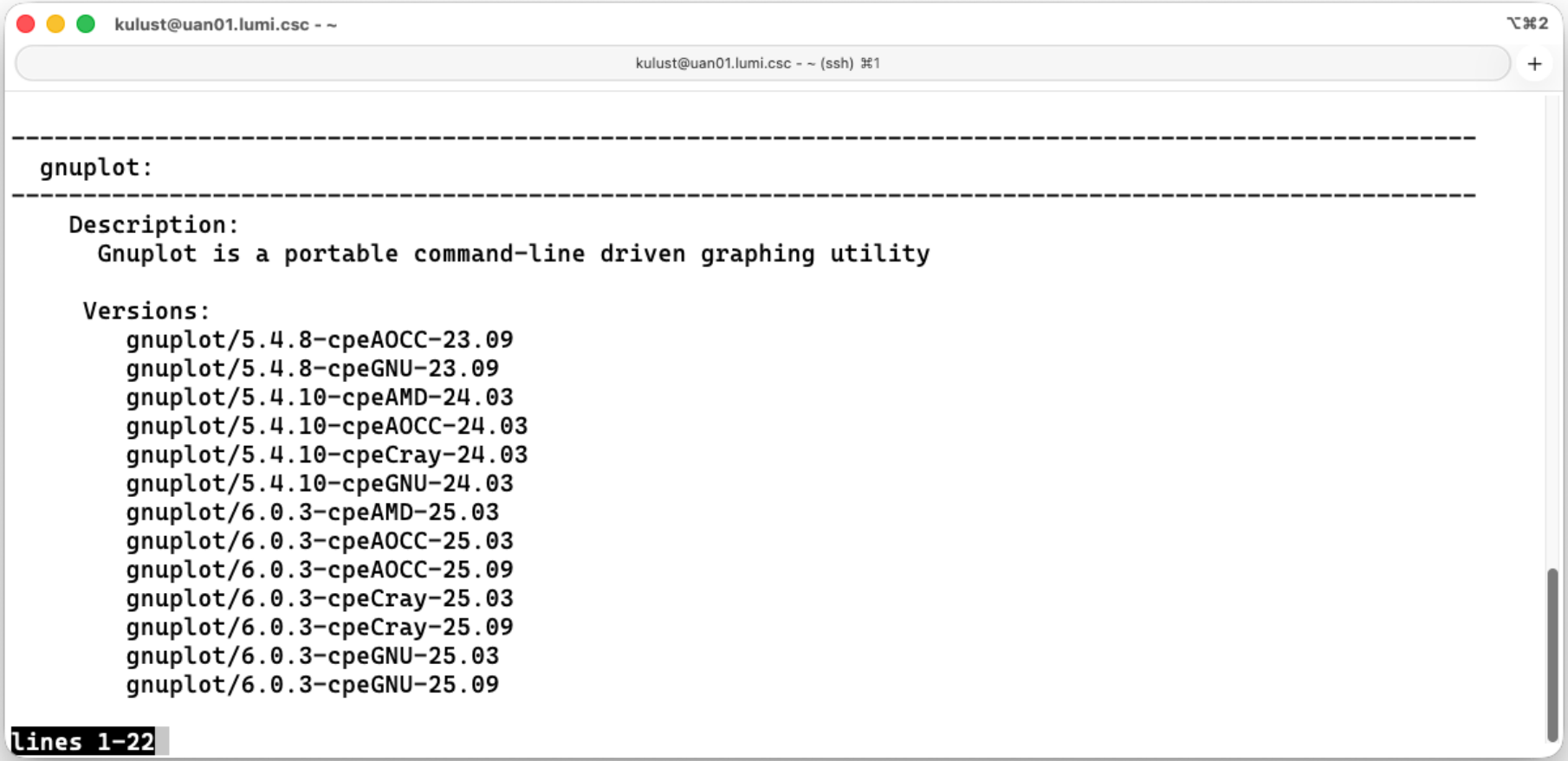
```
      LUMI/23.09  partition/D  craype-x86-rome
      LUMI/23.09  partition/D  craype-x86-spr
      LUMI/23.09  partition/D  craype-x86-trento
      LUMI/23.09  partition/D  craype-x86-turin
      LUMI/23.09  partition/G  craype-x86-genoa
      LUMI/23.09  partition/G  craype-x86-milan
      LUMI/23.09  partition/G  craype-x86-milan-x
      LUMI/23.09  partition/G  craype-x86-rome
      LUMI/23.09  partition/G  craype-x86-spr
      LUMI/23.09  partition/G  craype-x86-trento
      LUMI/23.09  partition/G  craype-x86-turin
      LUMI/23.09  partition/L  craype-x86-genoa
      LUMI/23.09  partition/L  craype-x86-milan
      LUMI/23.09  partition/L  craype-x86-milan-x
      LUMI/23.09  partition/L  craype-x86-rome
      LUMI/23.09  partition/L  craype-x86-spr
      LUMI/23.09  partition/L  craype-x86-trento
      LUMI/23.09  partition/L  craype-x86-turin

    Help:
      Documentation: `man intro_fftw3`

[lumi][kulust@uan01-1006 ~]$
```

module spider for a regular package

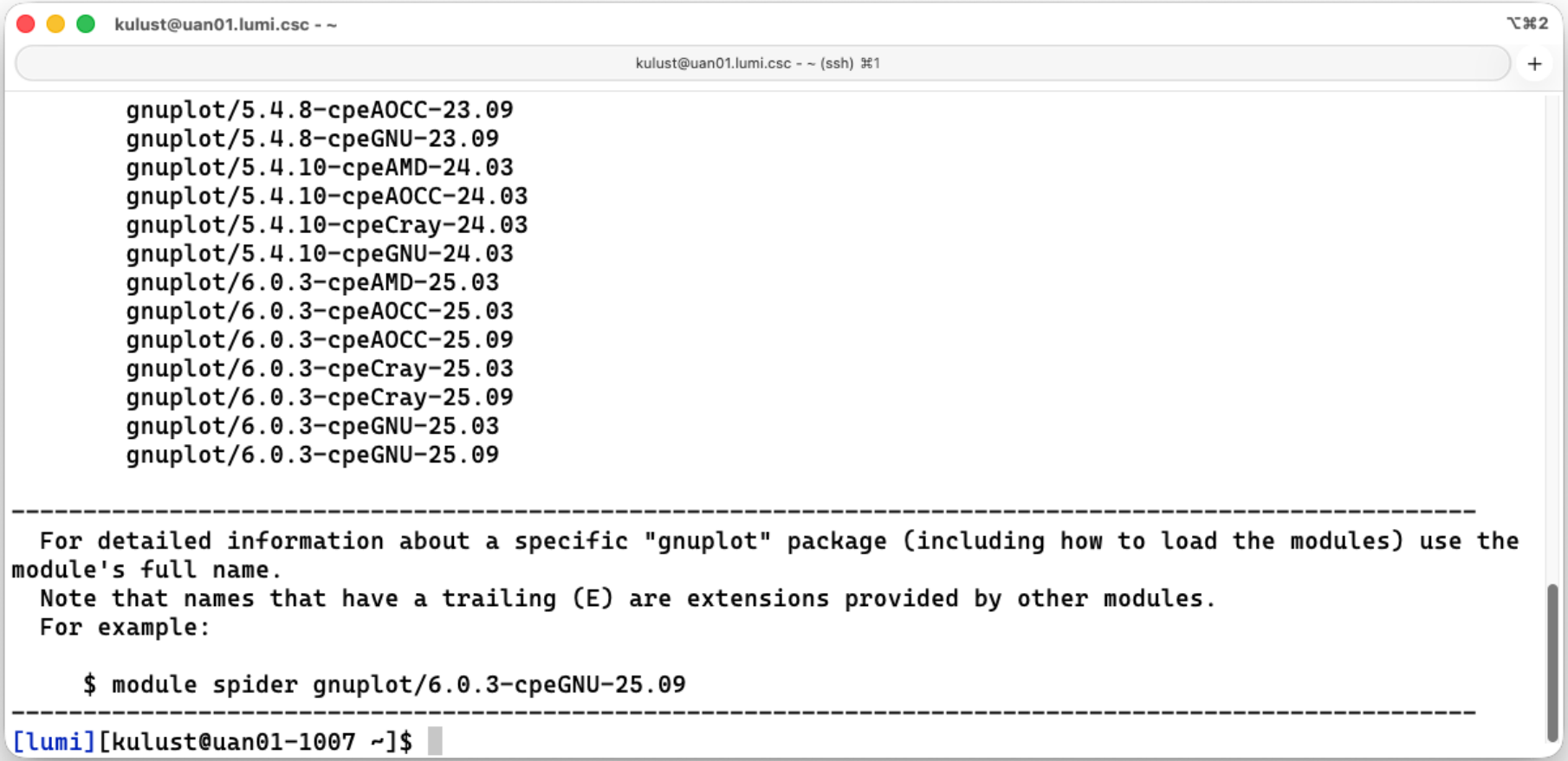
LUMI

- `module spider gnuplot` : Shows all versions of gnuplot on the system
- `module spider gnuplot/6.0.3-cpeGNU-25.09` : Shows help information for the specific module, including what should be done to make the module available

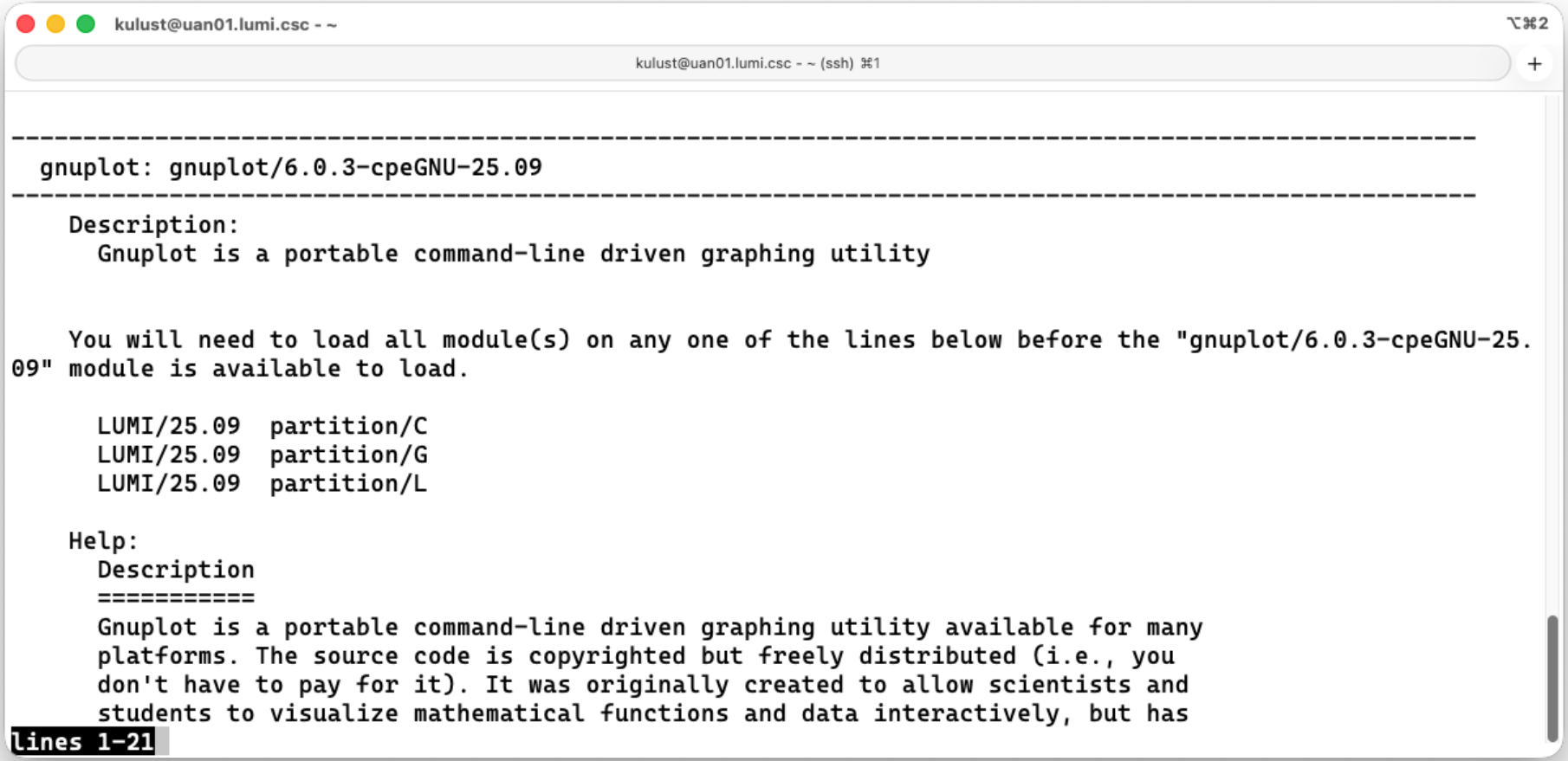
module spider gnuplot



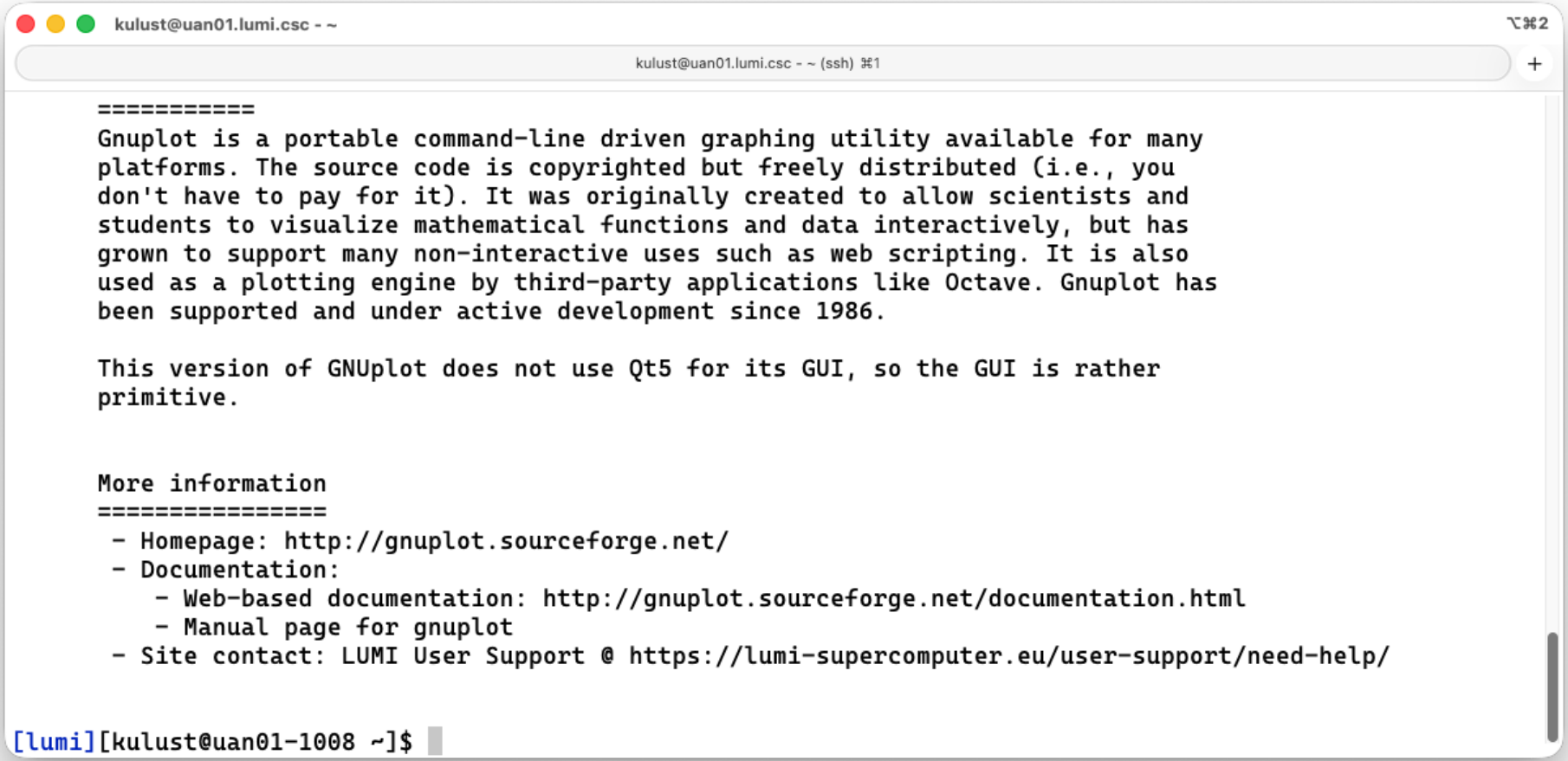
module spider gnuplot (2)



module spider gnuplot/6.0.3-cpeGNU-25.09



module spider gnuplot/6.0.3-cpeGNU-25.09 (2)

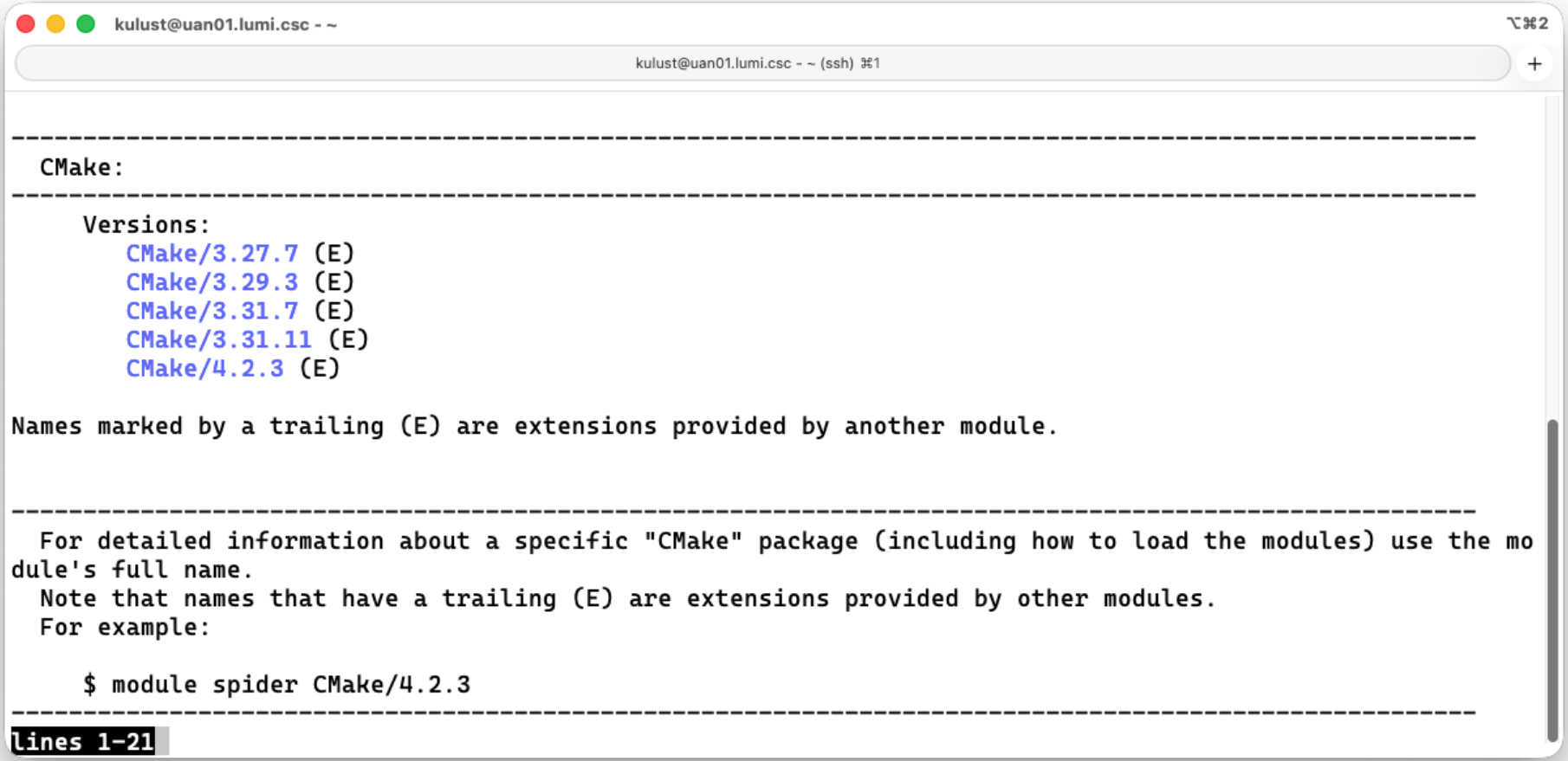


module spider for extensions

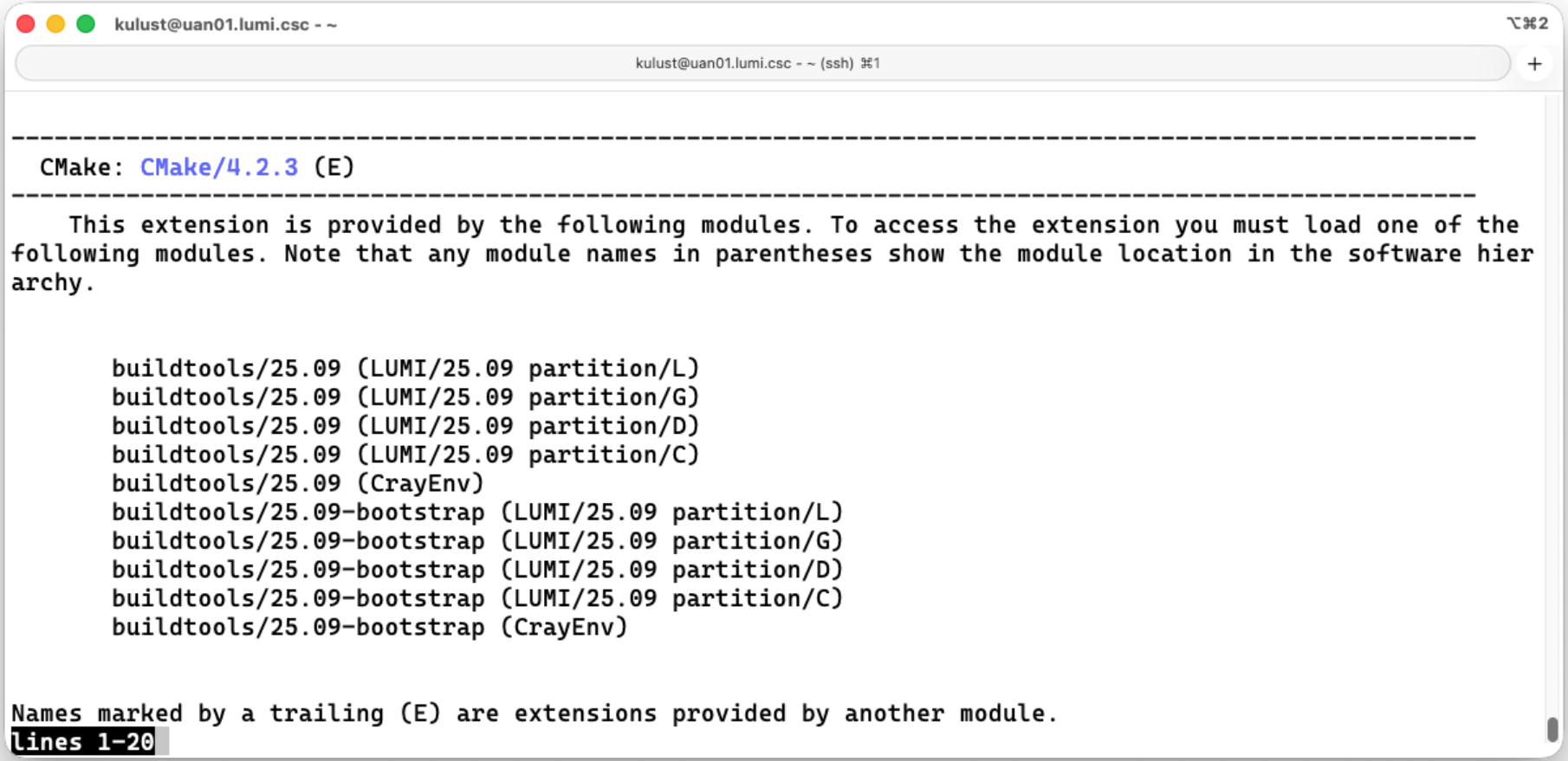
LUMI

- No example in the default Cray modules, so examples come from the LUMI software stacks
- `module spider CMake`
- `module spider CMake/4.2.3` : Will tell you which module contains this version of CMake and how to load it

module spider CMake



module spider CMake/4.2.3



module keyword

- Searches in the module short description and help message for the keyword.
  - E.g., try
    `module keyword editor`
- We do try to put enough information in the modules to make this a suitable additional way to discover software that is already installed on the system

module keyword editor

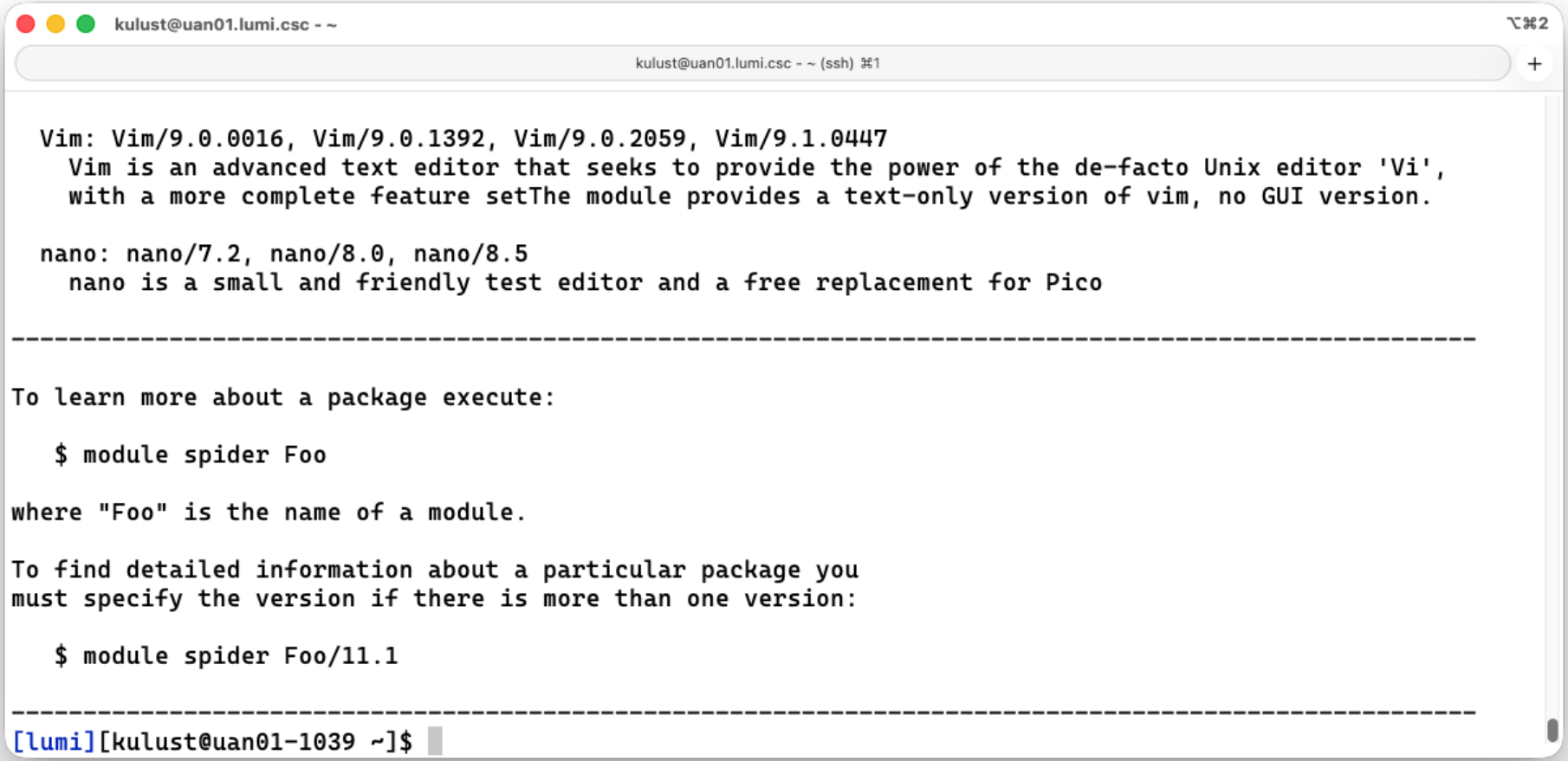
LUMI

kulust@uan01.lumi.csc - ~ ⌥⌘2

kulust@uan01.lumi.csc - ~ (ssh) ⌘1

```
------------------------------------------------------------------------------------------------------

The following modules match your search criteria: "editor"
------------------------------------------------------------------------------------------------------

  Vim: Vim/9.0.0016, Vim/9.0.1392, Vim/9.0.2059, Vim/9.1.0447
    Vim is an advanced text editor that seeks to provide the power of the de-facto Unix editor 'Vi',
    with a more complete feature setThe module provides a text-only version of vim, no GUI version.

  nano: nano/7.2, nano/8.0, nano/8.5
    nano is a small and friendly test editor and a free replacement for Pico

------------------------------------------------------------------------------------------------------

To learn more about a package execute:

   $ module spider Foo

where "Foo" is the name of a module.

To find detailed information about a particular package you
must specify the version if there is more than one version:
lines 1-22
```

module keyword editor (2)

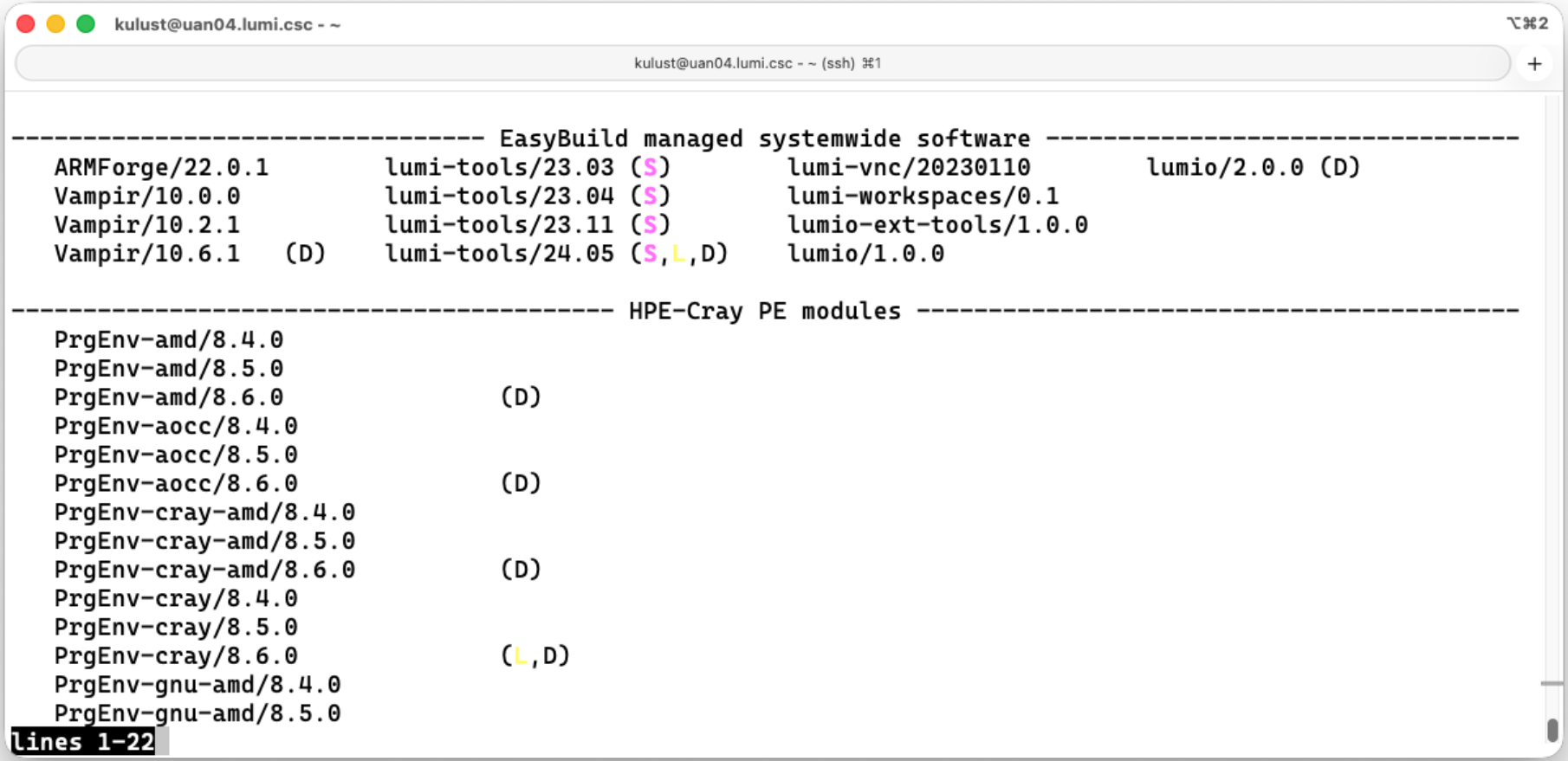


Sticky modules and module purge

LUMI

- On some systems, you will be taught to avoid `module purge` (which unloads all modules)
- Sticky modules are modules that are not unloaded by `module purge`, but reloaded.
  - They can be force-unloaded with `module --force purge` and `module --force unload` but use at your own risk!
- Used on LUMI for the software stacks and modules that set the display style of the modules
  - But keep in mind that the modules are reloaded which can have side effects
- And also used for some system initialisation (`init-lumi` and `lumi-tools`)

module av


kulust@uan04.lumi.csc - ~
kulust@uan04.lumi.csc - ~ (ssh) ⌘1
---------- EasyBuild managed systemwide software ----------
ARMForge/22.0.1 lumi-tools/23.03 (S) lumi-vnc/20230110 lumio/2.0.0 (D)
Vampir/10.0.0 lumi-tools/23.04 (S) lumi-workspaces/0.1
Vampir/10.2.1 lumi-tools/23.11 (S) lumio-ext-tools/1.0.0
Vampir/10.6.1 (D) lumi-tools/24.05 (S,L,D) lumio/1.0.0
---------- HPE-Cray PE modules ----------
PrgEnv-amd/8.4.0
PrgEnv-amd/8.5.0
PrgEnv-amd/8.6.0 (D)
PrgEnv-aocc/8.4.0
PrgEnv-aocc/8.5.0
PrgEnv-aocc/8.6.0 (D)
PrgEnv-cray-amd/8.4.0
PrgEnv-cray-amd/8.5.0
PrgEnv-cray-amd/8.6.0 (D)
PrgEnv-cray/8.4.0
PrgEnv-cray/8.5.0
PrgEnv-cray/8.6.0 (L,D)
PrgEnv-gnu-amd/8.4.0
PrgEnv-gnu-amd/8.5.0
lines 1-22

module av (2)

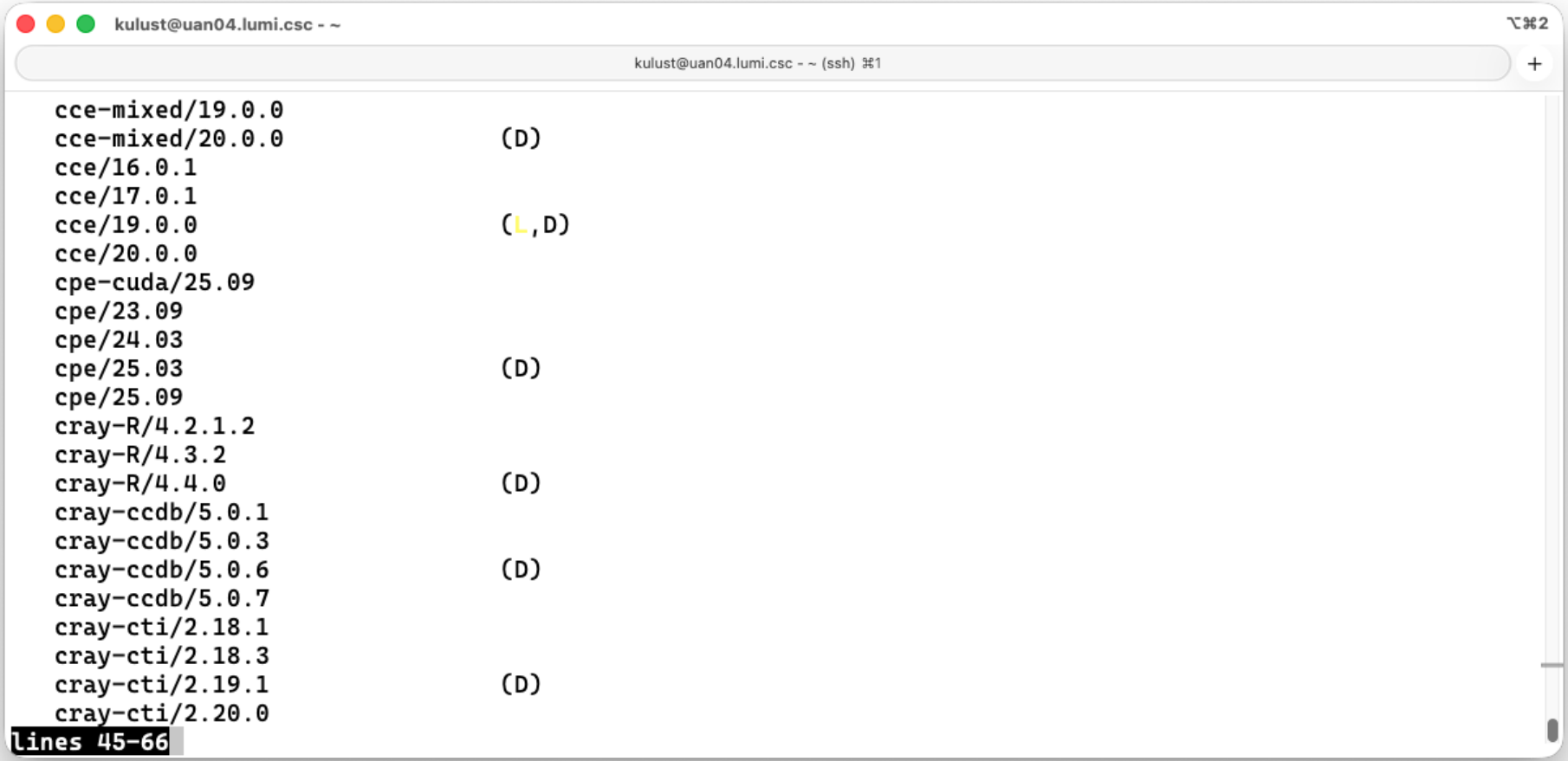
LUMI

kulust@uan04.lumi.csc - ~

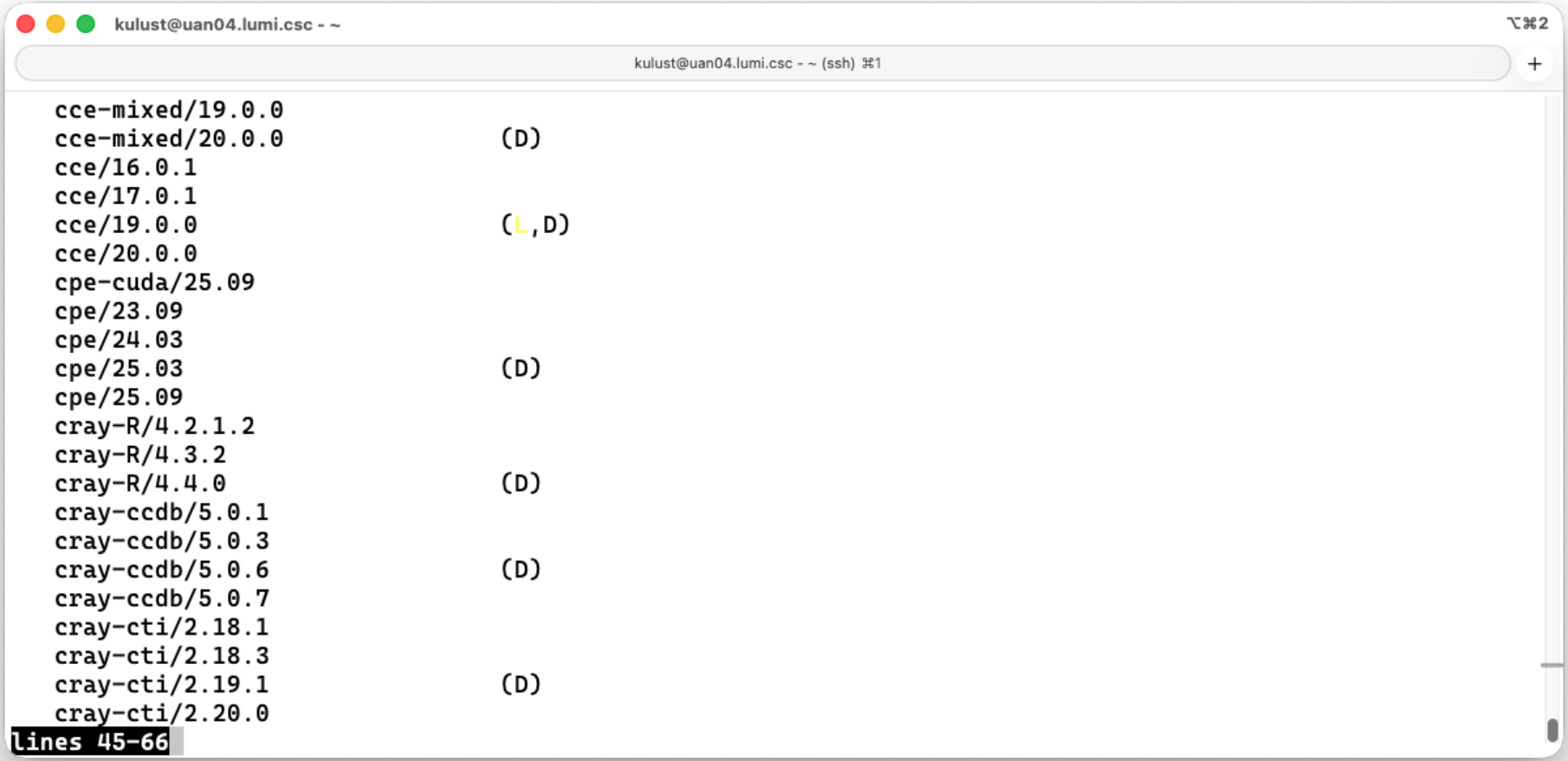
kulust@uan04.lumi.csc - ~ (ssh) ⌘1

```
   PrgEnv-gnu-amd/8.6.0          (D)
   PrgEnv-gnu/8.4.0
   PrgEnv-gnu/8.5.0
   PrgEnv-gnu/8.6.0              (D)
   PrgEnv-nvhpc/8.5.0
   PrgEnv-nvidia/8.5.0
   amd-mixed/6.3.4
   amd/6.3.4                     (5.0.2:5.1.0:5.2.0:5.2.3:5.5.1:5.7.0:6.0.0:6.0.3:6.2.1:6.3.0)
   aocc-mixed/3.2.0_module
   aocc-mixed/3.2.0
   aocc-mixed/4.1.0
   aocc-mixed/5.0.0              (D)
   aocc/3.2.0_module
   aocc/3.2.0
   aocc/4.1.0
   aocc/5.0.0                    (D)
   atp/3.15.1
   atp/3.15.3
   atp/3.15.6                    (D)
   atp/3.15.7
   cce-mixed/16.0.1
   cce-mixed/17.0.1
lines 23-44
```

module av (3)


kulust@uan04.lumi.csc - ~
⌥⌘2
kulust@uan04.lumi.csc - ~ (ssh) ⌘1
cce-mixed/19.0.0
cce-mixed/20.0.0 (D)
cce/16.0.1
cce/17.0.1
cce/19.0.0 (L,D)
cce/20.0.0
cpe-cuda/25.09
cpe/23.09
cpe/24.03
cpe/25.03 (D)
cpe/25.09
cray-R/4.2.1.2
cray-R/4.3.2
cray-R/4.4.0 (D)
cray-ccdb/5.0.1
cray-ccdb/5.0.3
cray-ccdb/5.0.6 (D)
cray-ccdb/5.0.7
cray-cti/2.18.1
cray-cti/2.18.3
cray-cti/2.19.1 (D)
cray-cti/2.20.0
lines 45-66

module av (4)


kulust@uan04.lumi.csc - ~
⌥⌘2
kulust@uan04.lumi.csc - ~ (ssh) ⌘1
cce-mixed/19.0.0
cce-mixed/20.0.0 (D)
cce/16.0.1
cce/17.0.1
cce/19.0.0 (L,D)
cce/20.0.0
cpe-cuda/25.09
cpe/23.09
cpe/24.03
cpe/25.03 (D)
cpe/25.09
cray-R/4.2.1.2
cray-R/4.3.2
cray-R/4.4.0 (D)
cray-ccdb/5.0.1
cray-ccdb/5.0.3
cray-ccdb/5.0.6 (D)
cray-ccdb/5.0.7
cray-cti/2.18.1
cray-cti/2.18.3
cray-cti/2.19.1 (D)
cray-cti/2.20.0
lines 45-66

module av (5)

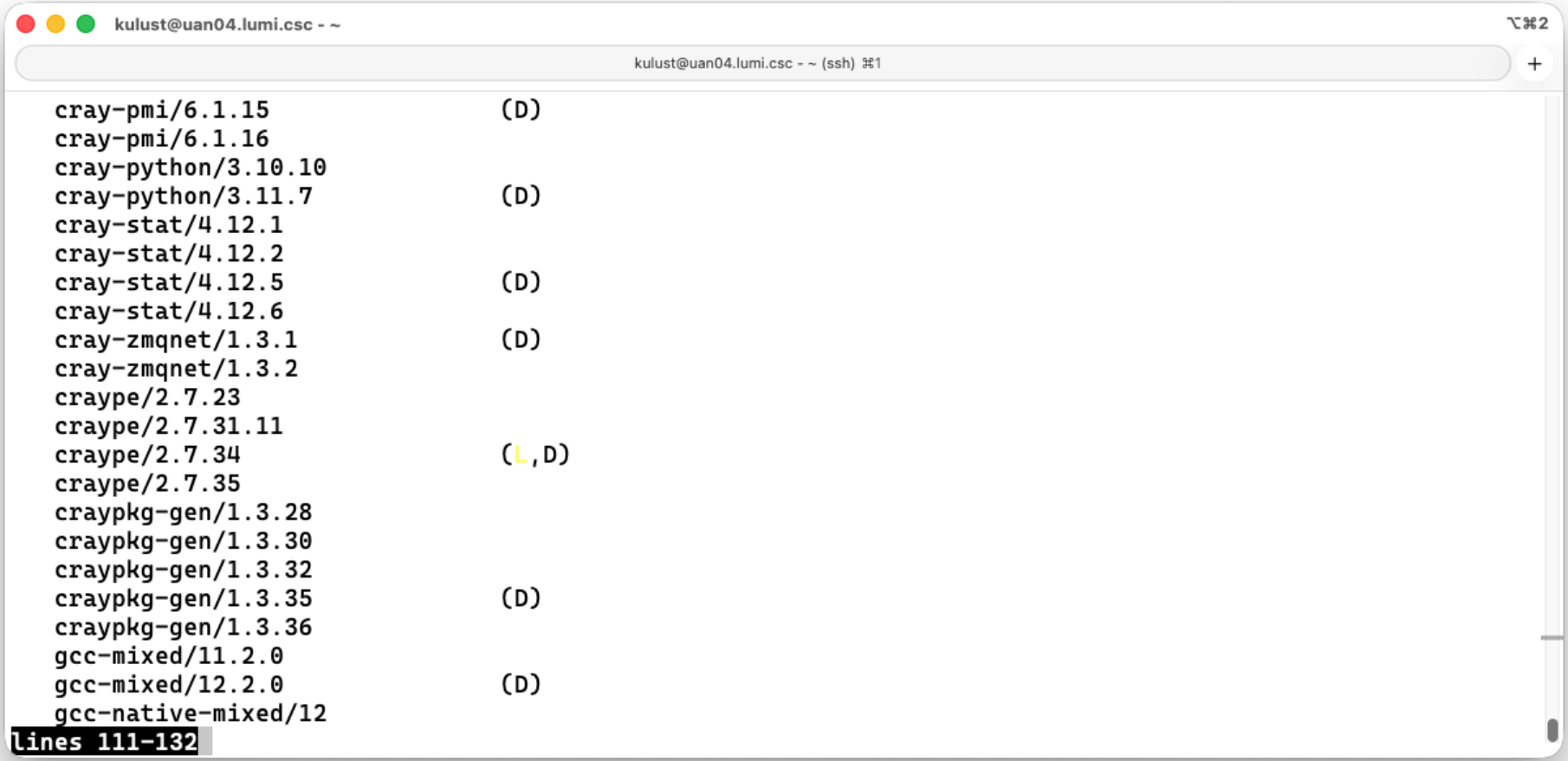
LUMI

kulust@uan04.lumi.csc - ~

kulust@uan04.lumi.csc - ~ (ssh) ⌘1

```
   cray-libsci_acc/25.09.2
   cray-mpich-abi/8.1.29
   cray-mpich-abi/8.1.32          (D)
   cray-mpich-abi/9.0.0
   cray-mpich/8.1.29
   cray-mpich/8.1.32              (L,D)
   cray-mpich/9.0.0
   cray-mpixlate/1.0.4
   cray-mpixlate/1.0.7            (D)
   cray-mrnet/5.1.1
   cray-mrnet/5.1.2
   cray-mrnet/5.1.5               (D)
   cray-mrnet/5.1.6
   cray-openshmemx/11.6.1
   cray-openshmemx/11.7.1
   cray-openshmemx/11.7.4         (D)
   cray-openshmemx/11.8.0
   cray-pals/1.2.12
   cray-parallel-netcdf/1.12.3.11 (D)
   cray-parallel-netcdf/1.12.3.17
   cray-pmi/6.1.12
   cray-pmi/6.1.14
lines 89-110
```

module av (6)


kulust@uan04.lumi.csc - ~
⌥⌘2
kulust@uan04.lumi.csc - ~ (ssh) ⌘1
cray-pmi/6.1.15 (D)
cray-pmi/6.1.16
cray-python/3.10.10
cray-python/3.11.7 (D)
cray-stat/4.12.1
cray-stat/4.12.2
cray-stat/4.12.5 (D)
cray-stat/4.12.6
cray-zmqnet/1.3.1 (D)
cray-zmqnet/1.3.2
craype/2.7.23
craype/2.7.31.11
craype/2.7.34 (L,D)
craype/2.7.35
craypkg-gen/1.3.28
craypkg-gen/1.3.30
craypkg-gen/1.3.32
craypkg-gen/1.3.35 (D)
craypkg-gen/1.3.36
gcc-mixed/11.2.0
gcc-mixed/12.2.0 (D)
gcc-native-mixed/12
lines 111-132

module av (7)

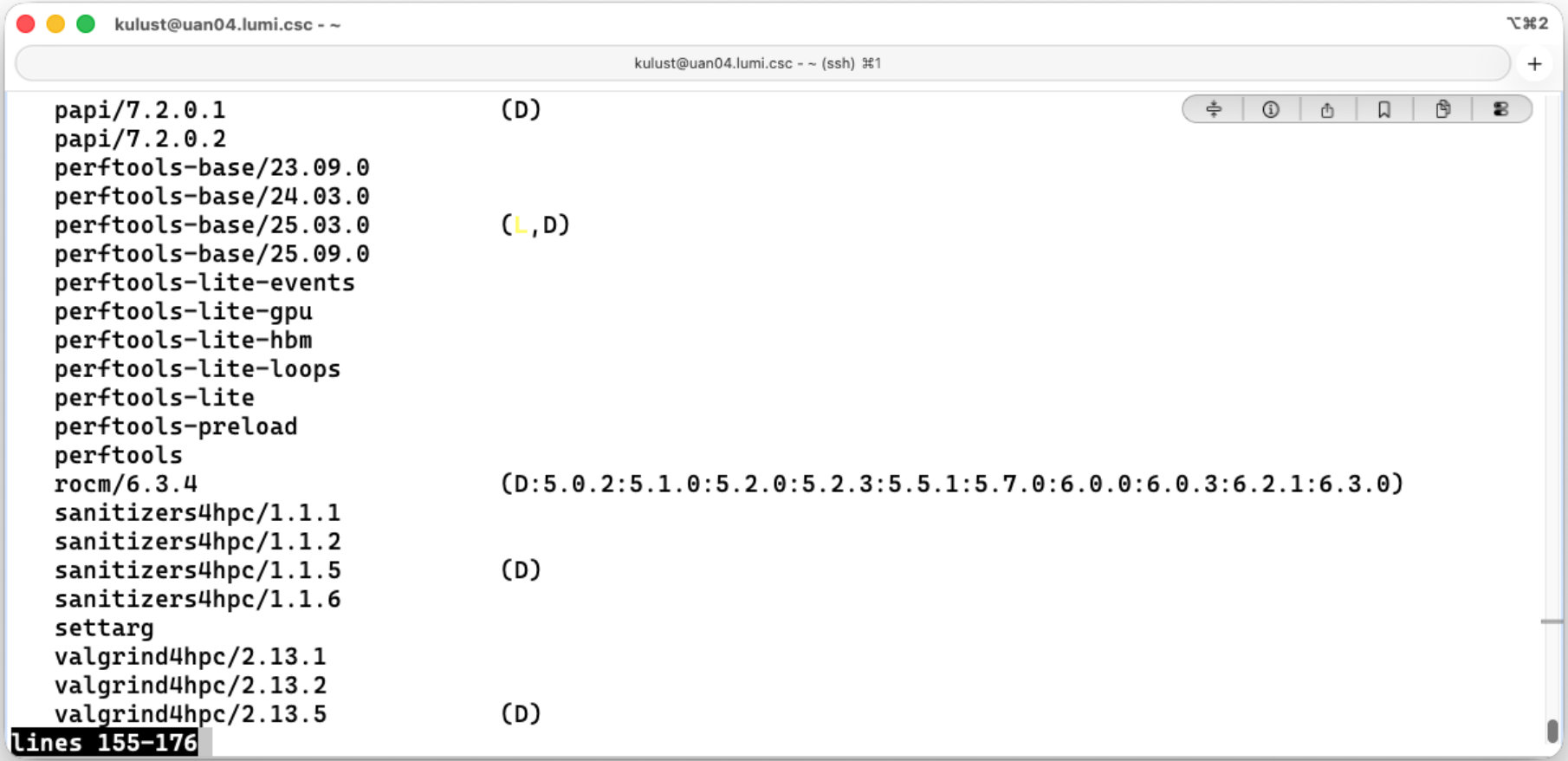
LUMI

kulust@uan04.lumi.csc - ~ ⌥⌘2

kulust@uan04.lumi.csc - ~ (ssh) ⌘1

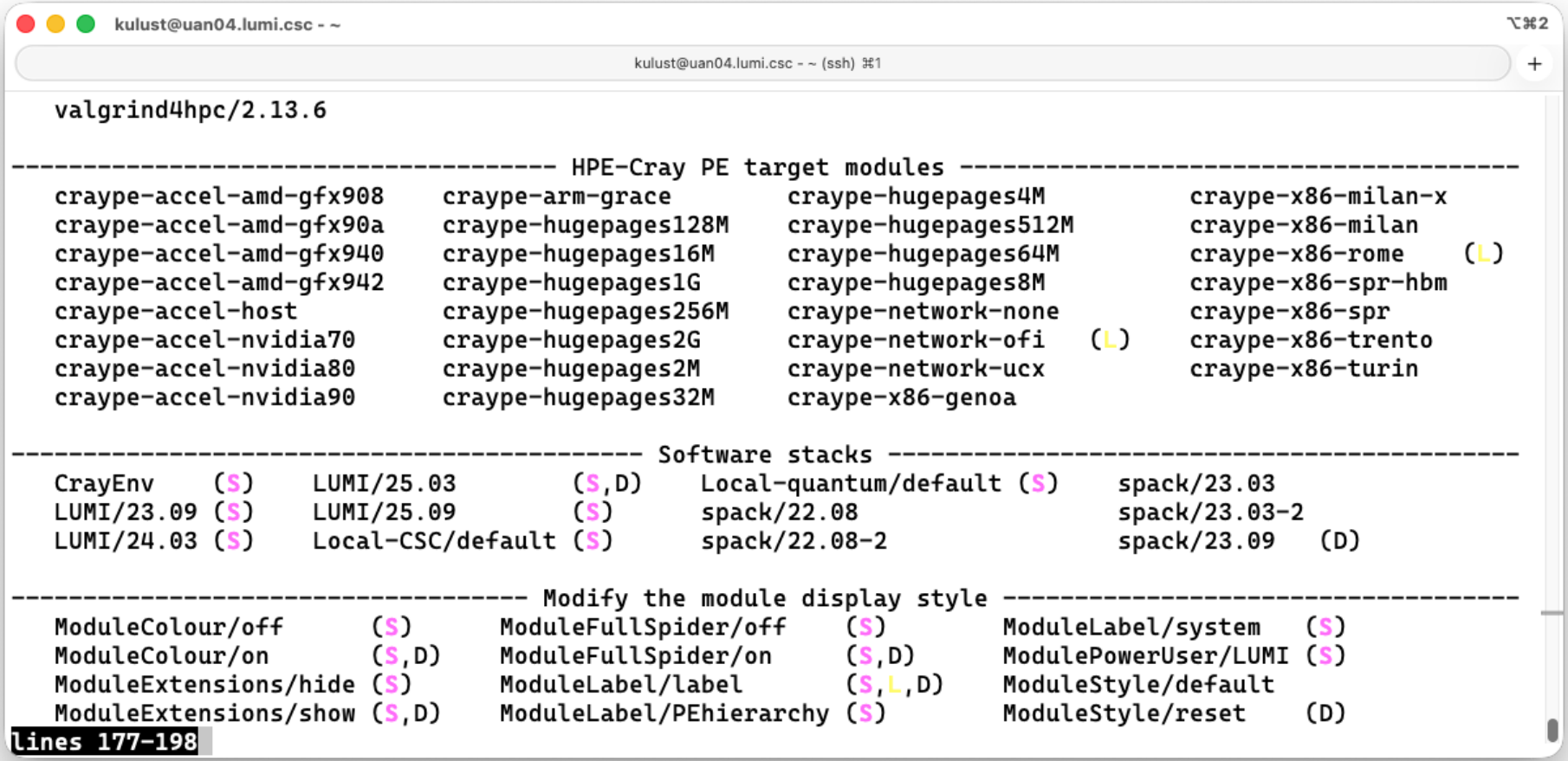
```
   gcc-native-mixed/12.3
   gcc-native-mixed/13
   gcc-native-mixed/13.2
   gcc-native-mixed/14
   gcc-native-mixed/14.2          (D)
   gcc-native/12
   gcc-native/12.3
   gcc-native/13
   gcc-native/13.2
   gcc-native/14
   gcc-native/14.2                (D)
   gcc/10.3.0
   gcc/11.2.0
   gcc/12.2.0                     (D)
   gdb4hpc/4.15.1
   gdb4hpc/4.16.1
   gdb4hpc/4.16.4                 (D)
   gdb4hpc/4.16.5
   iobuf/2.0.10
   lmod
   papi/7.0.1.1
   papi/7.1.0.1
lines 133-154
```

module av (8)


kulust@uan04.lumi.csc - ~
⌥⌘2
kulust@uan04.lumi.csc - ~ (ssh) ⌘1
papi/7.2.0.1 (D)
papi/7.2.0.2
perftools-base/23.09.0
perftools-base/24.03.0
perftools-base/25.03.0 (L,D)
perftools-base/25.09.0
perftools-lite-events
perftools-lite-gpu
perftools-lite-hbm
perftools-lite-loops
perftools-lite
perftools-preload
perftools
rocm/6.3.4 (D:5.0.2:5.1.0:5.2.0:5.2.3:5.5.1:5.7.0:6.0.0:6.0.3:6.2.1:6.3.0)
sanitizers4hpc/1.1.1
sanitizers4hpc/1.1.2
sanitizers4hpc/1.1.5 (D)
sanitizers4hpc/1.1.6
settarg
valgrind4hpc/2.13.1
valgrind4hpc/2.13.2
valgrind4hpc/2.13.5 (D)
lines 155-176

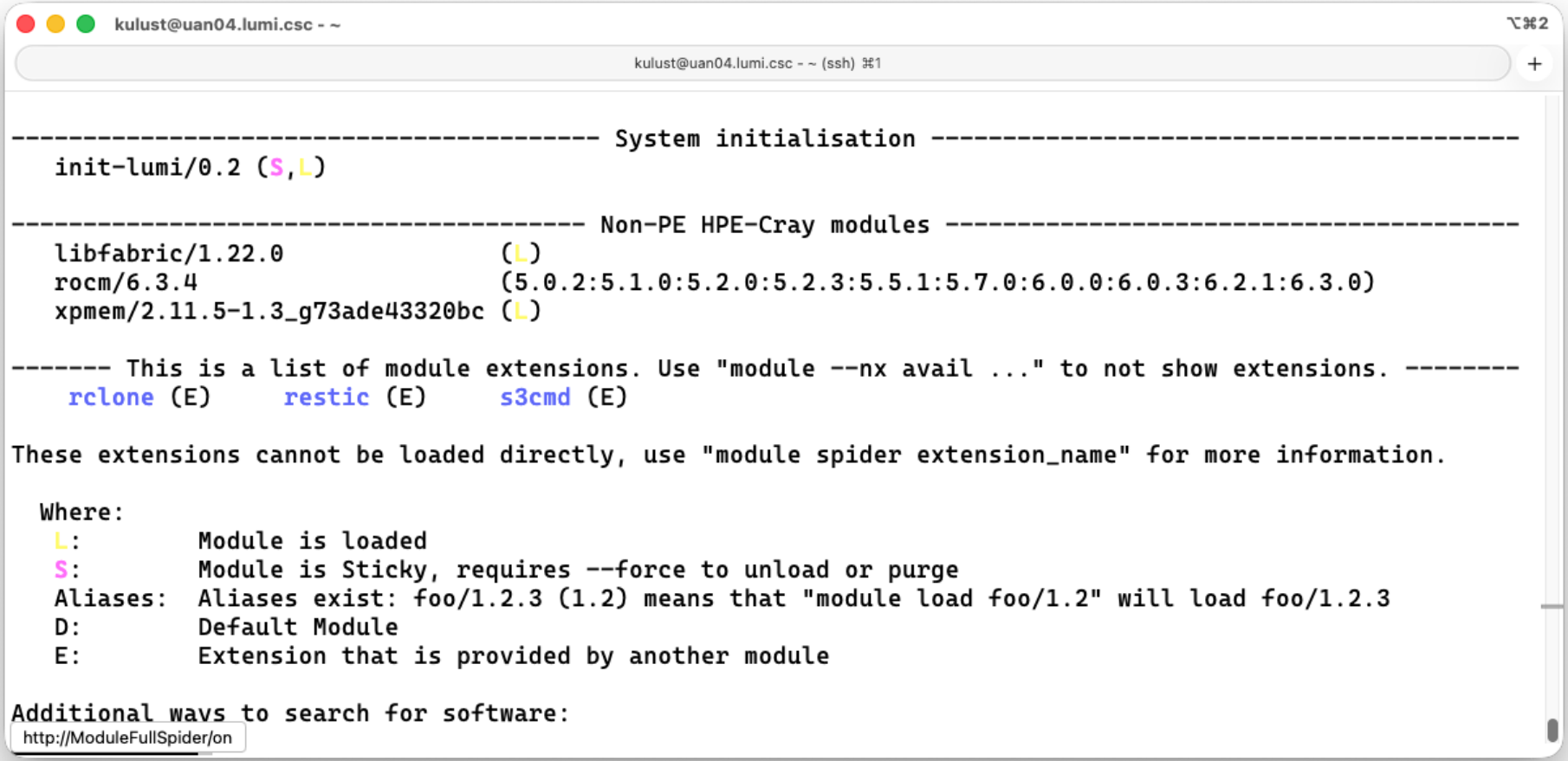
module av (9)

LUMI



module av (10)

LUMI



module av (11)

LUMI

kulust@uan04.lumi.csc - ~

kulust@uan04.lumi.csc - ~ (ssh) ⌘1

```
   xpmem/2.11.5-1.3_g73ade43320bc (L)

------- This is a list of module extensions. Use "module --nx avail ..." to not show extensions. --------
    rclone (E)     restic (E)     s3cmd (E)

These extensions cannot be loaded directly, use "module spider extension_name" for more information.

  Where:
   L:        Module is loaded
   S:        Module is Sticky, requires --force to unload or purge
   Aliases:  Aliases exist: foo/1.2.3 (1.2) means that "module load foo/1.2" will load foo/1.2.3
   D:        Default Module
   E:        Extension that is provided by another module

Additional ways to search for software:
* Use "module spider" to find all possible modules and extensions.
* Use "module keyword key1 key2 ..." to search for all possible modules matching any of the "keys".
See the LUMI documentation at https://docs.lumi-supercomputer.eu/runjobs/lumi_env/Lmod_modules/ for more i
nformation on searching modules.
If then you still miss software, contact LUMI User Support via https://lumi-supercomputer.eu/user-support/
need-help/.

[lumi][kulust@uan04-1002 ~]$
```

Changing how the module list is displayed

LUMI

- You may have noticed that you see descriptive texts in the module, not directories
- This can be changed by loading a module
  - `ModuleLabel/label` : The default view
  - `ModuleLabel/PEhierarchy` : Descriptive texts and unfolded PE hierarchy
  - `ModuleLabel/system` : Module directories
- Turn colour on or off using `ModuleColour/on` or `ModuleColour/off`
- Show or hide the module extensions with `ModuleExtensions/show` or `ModuleExtensions/hide`
- Index all modules with the spider command using `ModuleFullSpider/on`
- Show some hidden modules with `ModulePowerUser/LUMI`
  - This will also show undocumented/unsupported modules!
  - Can use `module --show_hidden avail` instead
- More customisation possible via LMOD environment variables

Getting help

LUMI

- `module help` is the command to get help information for available modules
  - Without further arguments: help about the module command
  - We do try to add a bit more help information about what a module provides to the modules than default EasyBuild or Spack installations tend to do.
- Examples (require loading `CrayEnv`):

```
module help cray-mpich
module help cray-python/3.11.7
module help buildtools/25.03
```

- `module whatis` can produce a short description

```
module whatis Subversion
module whatis Subversion/1.14.5
```

A note on caching

LUMI

- Large module system = lots of small module files = Lustre not very happy
  - But Lmod does use caches by default
  - Currently no system cache, only a user cache in `$HOME/.cache/lmod`
- Cache refreshed automatically every 24 hours
  - You'll notice when the `spider` or `available` commands are slow
  - But you may need to clean the cache after creating a new module as on LUMI Lmod does not always detect the change
- Also clear the cache if you notice very strange answers from `module spider`.
  - In the past, the HPE Cray PE has sometimes caused cache problems

A note on other commands

LUMI

- `module load`, `module unload`, `module list` are fairly standard commands and the basic operation is the same in all module systems
  - Note that `module list` may also show inactive modules: Modules that were loaded at some point but got unloaded when a module closer to the root of the hierarchy got unloaded
- `module swap`:
  - Equivalent to an unload followed by a load
  - For two modules of the same family `module swap` is more efficient as Lmod does not first have to discover the family conflict
  - But it is not essential as LUMI has autoswap enabled


LUMI
Questions?