Installing Software

Overview

While users cannot add or update system software or libraries using tools that require root privileges such as yum, many software packages can be installed locally without needing to be a superuser. Frequently linux packages make use of the "configure, make, make install" method and an example of how to do this is shown under Example Installation below.

Contrib Installations

One option to build your own software on HPC is to use the "contrib" environment. This is essentially a way to build your own module that you maintain and can share with other users. To create a contrib space, send a request to HPC Consult who will create a contrib space for your group. To see and use contrib software, use:

```
module load contrib
module load group_name/software_name
```

It should be noted that to see, access, and build software, you will need to be on one of the cluster's compute nodes. These can be accessed using the command `interactive`. More details can be found in our SLURM documentation.

When installing software, users are invited to use the compilers and libraries available through the module system (see previous section of this guide), but that is not necessary. If a software package you wish to install requires a specific version of the MPICH2 libraries, FFTW libraries, or even GCC compilers, you can install those packages for yourself.

Example Installation

- Software is not available on the login nodes. To install custom software, log into an interactive session.
- For a typical Linux installation, the default settings may attempt to install files in system directories outside your directories. This is not permitted, so the installation process (specifically, the "configure" step) needs to be changed so that files are installed in a specified location. There is frequently a *--prefix=/path/to/software* option.

Here is a typical example of installing software on a Linux cluster: Installing GROMACS (molecular simulation software):

1. **Get the software** I usually download a software package to my laptop, then transfer the downloaded package to my /home directory for installation. Alternatively, if I have the http or ftp address for the package I need, I can transfer that package directly to my /home directory while logged in using the wget utility:

   ```
   wget ftp://ftp.gromacs.org/pub/gromacs/gromacs-4.5.5.tar.gz
   ```

2. **Unzip and unpack the "tarball"**

   ```
   tar -zxf gromacs-4.5.5.tar.gz
   ```

3. **Read the instructions for installing the software**

   Here, the README file is a text file containing information and instructions, and it might have a different name like INSTALL or SETUP. Alternatively, the installation instructions may be posted on the software's website.

   ```
   cd gromacs-4.5.5
   more README
   ```

4. **Setup your environment for compiling the software**

   Here, to compile a parallel build of GROMACS, I need the MPICH2 libraries. Compiling GROMACS also requires some version of the GCC compilers and the FFTW libraries:

   ```
   module load gnu8/8.3.0 mpich/3.3.1 aocl-fftw/2.2
   ```

5. **Run the configure script**

   Typical Linux installations will make use of a script named "configure" that allows for customization. This step should be described in your software's installation instructions.

   ```
   ./configure --prefix=$HOME/gromacs/4.5.5 --disable-float --enable-mpi --without-x --disable-shared
   ```

   The `--prefix` options allows you to specify a local directory where the software should be installed and prevents the installation from trying to access system locations. In this example, a few other options are also used, such as the choice of not installing a graphical interface (--without-x).

6. **Run make to compile the source code**

   Typical Linux installations involve the process of compiling the software so it will run with the particular hardware and software available. The make command uses a special file called **Makefile** to guide the process of compiling large numbers of source code files (instead of just compiling one or a few source code files).

   ```
   make
   ```

7. **Install the newly compiled software**
   In this case, the executables, libraries, and other software files will be installed to the directory specified with the --prefix flag in step 5.

   ```
   make install
   ```

8. **Configure environment settings**
   Now that the new software package has been installed, you can customize your environment to give you access to your new software automatically. This usually involves appending the location of the new software to your existing environment variables, such as: **`PATH`**, **`LD_LIBRARY_PATH`**, **`INCLUDE`**. You may also need to add new environment variables that are specific to your software package. The installation instructions should explain what needs to be done. If you add these to the hidden file ~/.bashrc in your account, the environment variables will automatically be set each time you log in.

```
module add gcc/4.4 mpich2/1.4 fftw/2.1.5-double
export PATH=/home/u1/galen/gromacs/4.5.5/bin:$PATH
export LD_LIBRARY_PATH=/home/u1/galen/gromacs/4.5.5/lib:$LD_LIBRARY_PATH
export LIBRARY_PATH=/home/u1/galen/gromacs/4.5.5/lib:$LIBRARY_PATH
export C_INCLUDE_PATH=/home/u1/galen/gromacs/4.5.5/include:$C_INCLUDE_PATH
export CPLUS_INCLUDE_PATH=/home/u1/galen/gromacs/4.5.5/include:$CPLUS_INCLUDE_PATH
export MANPATH=/home/u1/galen/gromacs/4.5.5/share/man:$MANPATH
export MPI_HOME=/opt/mpich2/1.4
export FFTW_HOME=/opt/fftw/2.1.5-double
```