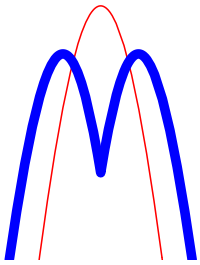
MOLPRO



Installation Guide
Version 2012.1

H.-J. Werner

Institut für Theoretische Chemie
Universität Stuttgart
Pfaffenwaldring 55
D-70569 Stuttgart
Federal Republic of Germany

P. J. Knowles

School of Chemistry
Cardiff University
Main Building, Park Place, Cardiff CF10 3AT
United Kingdom

SHA1 c4678c0bfb10df7a136fe8d2f70bbc43d96eab90

1 Obtaining the distribution materials

MOLPRO is distributed to licensees on a self-service basis using the world-wide web. Those entitled to the code should obtain it from https://www.molpro.net/download supplying the username and password given to them. The web pages contain both source code and binaries, although not everyone is entitled to source code, and binaries are not available for every platform.

Execution of MOLPRO, whether a supplied binary or built from source, requires a valid licence key. Note that the key consists of two components, namely a list of comma-separated key=value pairs, and a password string, and these are separated by '&'. In most cases the licence key will be automatically downloaded from the website when building or installing the software.

2 Installation of pre-built binaries

Binaries are given as self-extracting tar archives which are installed by running them on the command line. There are binaries tuned for several architectures. These also support parallel execution. The parallel binaries are built using GA with MPI. There is a generic serial binary which should run on all IA32 architectures.

The tar archives are fully relocatable, the location can be changed when running the script interactively, the default is `/usr/local`.

If the script finds a licence key which has been cached in `$HOME/.molpro/token` from a previous install then that key will be installed with the software. If the script cannot find a key or automatically download it from the molpro website then the script will prompt that this part of the install has failed. All files of Molpro are installed, but the user must then manually install the key with the library files in a file named `.token`, e.g.: `/usr/local/lib/molpro-`*`mpptype-arch`*`/lib/.t`

Other configuration options as described in section 3.5 may also be specified in the script file: `/usr/local/bin/molpro`

3 Installation from source files

3.1 Overview

There are usually four distinct stages in installing MOLPRO from source files:

Configuration — A shell script that allows specification of configuration options is run, and creates a configuration file that drives subsequent installation steps.

Compilation — The program is compiled and linked, and other miscellaneous utilities and files, including the default options file, are built. The essential resulting components are

1. The `molpro` shell script which launches thge main executable. In serial case one can directly run the main executable.
2. The `molpro.exe` executable, which is the main program. For parallel computation, multiple copies of `molpro.exe` are started by a single instance of `molpro` shell script using the appropriate system utility, e.g. `mpirun`, `parallel`, etc.

3. Machine-ready basis-set, and other utility, libraries.

Validation	A suite of self-checking test jobs is run to provide assurance that the code as built will run correctly.
Installation	The program can be run directly from the source tree in which it is built, but it is usually recommended to run the procedure that installs the essential components in standard system directories.

3.2 Prerequisites

The following are required or strongly recommended for installation from source code.

1. A Fortran 90 compiler. Fortran77-only compilers will not suffice. On HPC systems the latest vendor-supplied compiler should be used. The program is regularly tested with recent versions of GNU and Intel Fortran compilers.

2. GNU *make*, freely available from http://www.fsf.org and mirrors. GNU *make* must be used; most system-standard makes do not work. In order to avoid the use of a wrong *make*, it may be useful to set an alias, e.g., `alias make='gmake -s'`. A recent version of GNU *make* is required, 3.80 or above.

3. About 10GB disk space (strongly system-dependent; more with large-blocksize file systems, and where binary files are large) during compilation. Typically 100Mb is needed for the finally installed program. Large calculations will require larger amounts of disk space.

4. One or more large scratch file systems, each containing a directory that users may write on. There are parts of the program in which demanding I/O is performed simultaneously on two different files, and it is therefore helpful to provide at least two filesystems on different physical disks if other solutions, such as striping, are not available. The directory names should be stored in the environment variables `$TMPDIR`, `$TMPDIR2`, `$TMPDIR3`,.... These variables should be set before the program is installed (preferably in `.profile` or `.cshrc`), since at some stages the installation procedures will check for them (cf. section 3.5).

5. If the program is to be built for parallel execution then the Global Arrays toolkit or the MPI-2 library is needed. For building MOLPRO with the Global Arrays toolkit, we recommend the latest stable version (although earlier versions may also work). This is available from http://www.emsl.pnl.gov/docs/global and should be installed prior to compiling MOLPRO. For building MOLPRO with the MPI-2 library, we recommend to use the built-in MPI-2 library, which may have advantages of optimization on some platforms. If there is no built-in one on the platform, a fresh MPI-2 library (e.g.: MPICH2, see http://http://www.mpich.org/) should be installed prior to compiling MOLPRO. Many MPI-2 libraries, including Intel MPI, Bull MPI, MPICH2, and Open MPI, have been tested, and others untested could also work.

6. The source distribution of MOLPRO, which consists of a compressed tar archive with a file name of the form `molpro.2012.1.tar.gz`. The archive can be unpacked using `gunzip` and `tar`.

3.2.1 Fedora packages

To build using GNU compilers one should ensure the following packages are installed (via yum):

gcc-c++	provides GNU C and C++ compiler,
gcc-gfortran	provides GNU Fortran compiler,

Optionally one can choose to install:

blas-devel	provides a BLAS library,
lapack-devel	provides a LAPACK library,

which will be used instead of compiling the equivalent MOLPRO routines.

3.2.2 openSUSE packages

To build using GNU compilers one should ensure the following packages are installed (via YaST):

gcc-c++	provides GNU C and C++ compiler,
gcc-fortran	provides GNU Fortran compiler,
make	provides GNU make

Optionally one can choose to install:

blas	provides a BLAS library,
lapack	provides a LAPACK library,

which will be used instead of compiling the equivalent MOLPRO routines.

3.2.3 Ubuntu packages

To build using GNU compilers one should ensure the following packages are installed via (apt-get):

build-essential	provides GNU C++ compiler,
gfortran	provides GNU Fortran compiler,
curl	provides curl for downloading patches
openssh-server	provides ssh access to localhost

Optionally one can choose to install:

libblas-dev	provides a BLAS library,
liblapack-dev	provides a LAPACK library,

which will be used instead of compiling the equivalent MOLPRO routines. Set up password-less ssh by running the following commands and not entering a password when prompted:

```
ssh-keygen -t rsa
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

This must be done for each user account which will be running MOLPRO.

3.3 Configuration

Once the distribution has been unpacked, change to the `Molpro` directory that has been created. Having changed to the `Molpro` directory, you should check that the directory containing the Fortran compiler you want to use is in your PATH. Then run the command

```
./configure -batch
```

which creates the file `CONFIG`. This file contains machine-dependent parameters, such as compiler options. Normally `CONFIG` will not need changing, but you should at the least examine it, and change any configuration parameters which you deem necessary. Any changes made to `CONFIG` will be lost next time `./configure` is invoked, so it is best to supply as many of these as possible via the command line.

The `configure` procedure may be given command line options, and, if run without `-batch` , additionally prompts for a number of parameters:

1. On certain machines it is possible to compile the program to use either 32 or 64 bit integers, and in this case `configure` may be given a command-line option `-i4` or `-i8` respectively to override the default behaviour. Generally, the 64-bit choice allows larger calculations (files larger than 2Gb, more than 16 active orbitals), but can be slower if the underlying hardware does not support 64-bit integers. Note that if `-i4` is used then large files (greater than 2Gb) are supported on most systems, but even then the sizes of MOLPRO records are restricted to 16 Gb since the internal addressing in MOLPRO uses 32-bit integers. If `-i8` is used, the record and file sizes are effectively unlimited. Normally we recommend using the default determined by configure.

2. In the case of building for parallel execution, the option `-mpp` must be given on the command line. This enables both mpp and mppx parallelism; for the distinction between these two parallelism modes, please refer to the user manual, section 2. The option `-mppbase` must also be given followed by the location of the Global Arrays build directory or the MPI-2 library include directory.

   For the case of using the Global Arrays toolkit, one example can be

   ```
   ./configure -mpp -mppbase /usr/local/ga-[version]
   ```

   If using a Global Arrays build with an MPI library the appropriate MPI executable should appear first in PATH when more than one is available.

   Queries regarding Global Arrays installations should be sent directly to the Global Arrays team, any Molpro related queries will assume a fully functional Global Arrays suite with all internal tests run successfully.

   For the case of using the MPI-2 library, one example can be

   ```
   ./configure -mpp -mppbase /usr/local/mpich2-install/include
   ```

   and the `-mppbase` directory should contain file mpi.h. Please ensure the built-in or freshly built MPI-2 library fully supports MPI-2 standard and works properly.

   For desktop or single node installations, there are a series of options prefixed with `-auto` which build any prerequisites, and can be used in place of `-mppbase`, eg.

   ```
   ./configure -mpp -auto-ga-mpich
   ```

3. If any system libraries are in unusual places, it may be necessary to specify them explicitly as the arguments to a `-L` command-line option.

4. `configure` asks whether you wish to use system BLAS subroutine libraries. MOLPRO has its own optimised Fortran version of these libraries, and this can safely be used. On most machines, however, it will be advantageous to use a system-tuned version instead. On the command line one can specify the level of BLAS to be used from the system, e.g. `-blas2`. For example if you specify 2, the system libraries will be used for level 2 and level 1 BLAS, but MOLPRO's internal routines will be used for level 3 (i.e., matrix-matrix multiplication). Normally, however, one would choose either 0 or 3, which are the defaults depending upon whether a BLAS library is found.

   A special situation arises if 64-bit integers are in use (-i8), since on many platforms the system BLAS libraries only supports 32-bit integer arguments. In such cases (e.g., IBM, SGI, SUN) either 0 or 4 can be given for the BLAS level. BLAS=0 should always work and means that the MOLPRO Fortran BLAS routines are used. On some platforms (IBM, SGI, SUN) BLAS=4 will give better performance; in this case some 32-bit BLAS routines are used from the system library (these are then called from wrapper routines, which convert 64 to 32-bit integer arguments. Note that this might cause problems if more than 2 GB of memory is used).

   For good performance it is important to use appropriate BLAS libraries; in particular, a fast implementation of the matrix multiplication `dgemm` is very important for MOLPRO. Therefore you should use a system tuned BLAS library whenever available.

   MOLPRO will automatically detect the most appropriate BLAS library in many cases. In certain cases, in particular when the BLAS library is installed in a non-default location, configure should be directed to the appropriate directory with:

```
./configure -blaspath /path/to/lib/dir
```

   Specification of BLAS libraries can be simplified by placing any relevant downloaded libraries in the directory `blaslibs`; `configure` searches this directory (and then, with lower priority, some potential system directories) for libraries relevant to the hardware.

   For Intel and AMD Linux systems we recommend the following BLAS libraries:

MKL	The Intel Math Kernel Library (MKL)
ATLAS	The Automatically Tuned Linear Algebra Software (ATLAS) library. You must use the atlas library specific to your processor:

Pentium III	`Linux_PIIISSE1`
Pentium 4,Xeon	`Linux_P4SSE2`
AMD Athlon	`Linux_ATHLON`
AMD Opteron	`Linux_HAMMER64SSE2_2` (64 bit)

	When using atlas MOLPRO will automatically compile in the extra lapack subroutines which do not come by default with the package and so the liblapack.a which comes with Atlas is sufficient.
ACML	For Opteron systems then AMD Core Math Library (ACML) is the preferred blas library.

   SGI Altix can use the `scsl` library is preferred. HP platforms can use the `mlib` math library. IBM Power platforms can use the `essl` package.

5. `configure` prompts for the optional bin directory (`INSTBIN`) for linking MOLPRO. This directory should be one normally in the PATH of all users who will access MOLPRO, and its specification will depend on whether the installation is private or public.
6. `configure` prompts for the Molpro installation directory (`PREFIX`).
7. `configure` prompts for the destination directory for documentation. This should normally be a directory that is mounted on a worldwide web server. This is only relevant if the documentation is also going to be installed from this directory (see below).

The full list of command-line options recognized by `configure` are:

`-af90`	use Absoft Pro Fortran compiler
`-auto-ga-hpmpi`	auto-build GA with MPI and HP MPI
`-auto-ga-mpich`	auto-build GA with MPI and MPICH
`-auto-ga-mvapich2ib`	auto-build GA with MPI and MVAPICH2 over Infiniband
`-auto-ga-openmpi`	auto-build GA with MPI and Open MPI
`-auto-ga-openmpi-sge`	auto-build GA with MPI and Open MPI with SGE support
`-auto-mpich`	auto-build MPICH
`-auto-mvapich2ib`	auto-build MVAPICH2 over Infiniband
`-auto-openmpi`	auto-build Open MPI
`-auto-openmpi-sge`	auto-build Open MPI with SGE support
`-batch`	run script non-interactively
`-blas`	use external BLAS library
`-blaspath`	specify blas library path
`-Block`	compile Block code
`-cc`	use C compiler named cc
`-clang`	use Clang C compiler
`-cuda`	try to get settings for compiling CUDA code
`-f90`	use f90 Fortran compiler
`-fcc`	use Fujitsu C compiler
`-force-link`	Force linking of main executable
`-fort`	use fort Fortran compiler
`-frt`	use frt Fortran compiler
`-g95`	use G95 Fortran compiler
`-gcc`	use GNU Compiler Collection C compiler
`-gforker`	Use settings for mpich2 configured with gforker option
`-gfortran`	use gfortran Fortran compiler
`-i386`	use settings for i386 machine
`-i4`	Makes default integer variables 4 bytes long
`-i686`	use settings for i686 machine
`-i8`	Makes default integer variables 8 bytes long
`-icc`	use Intel C compiler

`-ifort`	use Intel Fortran compiler
`-inst-pl`	append PL to PREFIX when running make install
`-intel-mpi-lsf`	Use settings for Intel MPI with LSF
`-j`	number of make threads for building prerequisites
`-lapack`	use external LAPACK library
`-lapackpath`	specify LAPACK library path
`-letter`	specify letter latex paper size
`-mpp`	produce parallel Molpro
`-mppbase`	specify mpp base path for includes and libraries
`-nagfor`	use NAG Fortran compiler
`-natom`	max number of atoms
`-nbasis`	max number of basis functions
`-noaims`	do not compile aims code
`-noblas`	Don't use external BLAS library
`-noboost`	Do not use binary part of Boost library
`-nocuda`	don't compile CUDA code
`-nocxx`	do not compile C++ code
`-nolapack`	don't use external LAPACK library
`-nolargefiles`	Do not use largefiles
`-noneci`	do not compile neci code
`-noopenmp`	compile without openmp
`-noxml2`	do not use libxml2
`-nprim`	max number of primitives
`-nrec`	max number of records
`-nstate`	max number of states per symmetry
`-nsymm`	max number of state symmetries
`-nvalence`	max number of valence orbitals
`-nvcc`	use NVIDIA CUDA C compiler
`-opencc`	use Open64 C compiler
`-openf90`	use Open64 Fortran compiler
`-openmp`	compile with openmp
`-openmp-mismatch`	Override exit with mismatched compilers
`-openmpi`	Use settings for standard openmpi
`-openmpi-sge`	Use settings for openmpi compiled with SGE
`-pathcc`	use Pathscale C compiler
`-pathf90`	use Pathscale Fortran compiler
`-pgcc`	use Portland C compiler
`-pgf90`	use Portland Fortran compiler
`-prefix`	Specify top-level installation directory

-slater	compile slater code
-sm_13	Use settings for sm_13 architecture for CUDA compilation
-sm_20	Use settings for sm_20 architecture for CUDA compilation
-suncc	use Sun C compiler
-sunf90	use Sun Fortran compiler
-x86_64	use settings for 64-bit x86 machine
-xlc	use IBM compiler
-xlf	use IBM Fortran compiler

3.4 Compilation and linking

After configuration, the remainder of the installation is accomplished using the GNU *make* command. Remember that the default *make* on many systems will not work, and that it is essential to use GNU *make* (cf. section 3.2). Everything needed to make a functioning program together with all ancillary files is carried out by default simply by issuing the command

```
make
```

in the MOLPRO base directory. Most of the standard options for GNU *make* can be used safely; in particular, `-j` can be used to speed up compilation on a parallel machine. The program can then be accessed by making sure the `bin/` directory is included in the PATH and issuing the command `molpro`. If MPI library is used for building Global Arrays or building MOLPRO directly, please be aware that some MPI libraries use mpd daemons to launch parallel jobs. In this case, mpd daemons must already be running before `make`.

3.5 Adjusting the default environment for MOLPRO

The default running options for MOLPRO are stored in the script `bin/molpro`. After program installation, either using binary or from source files, this file should be reviewed and adjusted, if necessary, to make system wide changes.

3.6 Tuning

MOLPRO can be tuned for a particular system by running in the root directory the command

```
make tuning
```

This job automatically determines a number of tuning parameters and appends these to the file `bin/molpro`. Using these parameters, MOLPRO will select the best BLAS routines depending on the problem size. This job should run on an empty system. It may typically take 10 minutes, depending on the processor speed, and you should wait for completion of this run before doing the next steps.

3.7 Testing

At this stage, it is essential to check that the program has compiled correctly. The makefile target *test* (i.e., command `make test`) will do this using the full suite of test jobs, and although this takes a significantly long time, it should always be done when porting for the first time. A much faster test, which checks the main routes through the program, can be done using `make`

`quicktest`. For parallel installation, it is highly desirable to perform this validation with more than one running process. This can be done conveniently through the `make` command line as, for example,

```
make MOLPRO_OPTIONS=-n2 test
```

If any test jobs fail, the cause must be investigated. If, after due efforts to fix problems of a local origin, the problem cannot be resolved, the developers of MOLPRO would appreciate receiving a report. There is a web-based mechanism at https://www.molpro.net/bugzilla at which as many details as possible should be filled in. It may also be helpful to attach a copy of the CONFIG file along with the failing output. Please note that the purpose of such bug reports is to help the developers improve the code, and not for providing advice on installation or running.

3.8 Installing the program for production

Although the program can be used in situ, it is usually convenient to copy only those files needed at run time into appropriate installation directories as specified at configuration time (see section 3.3) and stored in the file `CONFIG`. To install the program in this way, do

```
make install
```

The complete source tree can then be archived and deleted. The overall effect of this is to create a shell script in the `INSTBIN` directory. The name should relate to the architecture, type of build, integer etc. Symbolic links relating to the type of build are then made, and finally providing that `INSTBIN/molpro` is not a file, a symbolic link is created to the new script. In some cases it is preferable to create a localized script in `INSTBIN/molpro` which will not be over written. The overall effect of this cascade of links is to provide, in the normal case, the commands `molpro` and one or both of `molpros` (serial) and `molprop` (parallel) for normal use, with the long names remaining available for explicit selection of particular variants.

For normal single-variant installations, none of the above has to be worried about, and the `molpro` command will be available from directory `INSTBIN`.

During the install process the key from `$HOME/.molpro/token` is copied to `PREFIX/.token` so that the key will work for all users of the installed version.

3.9 Installation of documentation

The documentation is available on the web at http://www.molpro.net/info/users. It is also included with the source code. The PDF user's manual is found in the directory `Molpro/doc/manual.pdf`, with the HTML version in the directory `Molpro/doc/manual/index.html`. After `make install` the documentation is installed in the `doc` subdirectory of `PREFIX` specified in `CONFIG` file generated by the `configure` command. Numerous example input files are included in the manual, and can alternatively be seen in the directory `Molpro/examples`.

3.10 Simple building for single workstations Linux or Mac OS X

The following instructions are quick instructions for installing MOLPRO on a single-workstation Linux or Mac OS X system. The instructions assume GNU compilers have been installed (details of getting GNU compilers for common Linux distributions are contained in the prerequisites section), but these can be substituted with alternative compilers. For serial MOLPRO:

```
./configure -batch -gcc -gfortran
make
```

For parallel MOLPRO one can use:

```
./configure -batch -gcc -gfortran -mpp -mppbase /path/to/ga/build
make
```

if Global Arrays has already been built. There is a simpler option, providing the curl utility is installed, and the machine is connected to the internet:

```
./configure -batch -gcc -gfortran -mpp -auto-ga-mpich
make
```

which will automatically download and install MPICH and Global Arrays.

3.11 Installation on a Cygwin system

On a Windows machine Cygwin should be installed. In addition to the default package list one should also install the packages listed in table 1. If undertaking development work table 2

Package	Package Group	Reason
gcc-core	Devel	compiling C files
gcc-fortran	Devel	compiling Fortran files
gcc-g++	Devel	compiling C++ files
make	Devel	need GNU make
ca-certificates	Net	download boost
curl	Net	token download
libgmp3	Libs	bug? needed by make.exe
libltdl7	Devel	bug? needed by make.exe

Table 1: Cygwin requirments for user install

contains a list of potentially useful packages.

Package	Package Group	Reason
bison	Devel	bison
gdb	Devel	gdb
git	Devel	git
libxslt	Libs	xsltproc
openssh	Net	ssh
vim	Editors	vi

Table 2: Cygwin packages for developers

With the above steps, configure can be run and the Molpro built in the normal way.