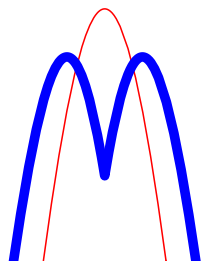


# MOLPRO



## *Installation Guide Version 2010.1*

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 833379180250ebe11052d872bd9ff53639690c67*

(Copyright ©2008 University College Cardiff Consultants Limited)

## 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 TCGMSG, and the default connection across nodes is rsh. One can use ssh instead of rsh by changing environment variable TCGRSH (e.g., export TCGRSH=/usr/bin/ssh for bash). 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/.token`

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 <ol style="list-style-type: none"><li>1. The <code>molpro</code> shell script which launches the main executable. In serial case one can directly run the main executable.</li><li>2. The <code>molpro.exe</code> executable, which is the main program. For parallel computation, multiple copies of <code>molpro.exe</code> are started</li></ol>

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 most systems, the latest vendor-supplied compiler should be used.

For IA32 Linux (for example Intel Pentium or AMD athlon) the recommended compilers are the Intel Compiler `ifort` version 10.1 or higher or the Portland `pgf90` compiler version 7.2 or higher. For Opteron and EM64T systems the recommended compilers are Portland version 7.2 or higher, Pathscale compiler `pathf90` version 3.1 or higher, or the Intel Compiler version 10.1 or higher. The full list of supported compilers can be found at <http://www.molpro.net/supported>.

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. The GNU *curl* utility for batch-mode http transfers, although not needed for installation, is essential for any subsequent application of patches that implement bug fixes.
4. 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.
5. 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).
6. 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. As mentioned in Global Arrays documentation, there are three possible ways for building GA: (1) GA with MPI; (2) GA with TCGMSG-MPI; and (3) GA with TCGMSG. Molpro can work with either of these interfaces. 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://www.mcs.anl.gov/research/projects/mpich2> ) 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.

7. The source distribution of MOLPRO, which consists of a base compressed tar archive with a file name of the form `molpro.2010.1.tar.gz`, together, possibly, with one or more *module* archives with file names of the form `molpro.module.2010.1.tar.gz`. The modules contain code which is not generally distributed, or features which are not always required to install the code. An example is the program developers' kit (*module=develop*). The archives can be unpacked using `gunzip` and `tar`. All archives must be unpacked in the same directory. It is essential that the base archive is unpacked first, and advisable that any modules are unpacked before further installation.

### 3.3 Configuration

Once the distribution has been unpacked, identify the root directory that was created (normally `molpro2010.1`). In the following description, all directories are given relative to this root. Having changed to the root 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` or `-mppx` must be given on the command line. 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 base 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-4-3-3
```

and the `-mppbase` directory should contain directories `include` and `lib/TARGET` where for example on Linux/x86\_64 `TARGET=LINUX64`.

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.

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 `mllib` 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
-aims         compile aims code
-auto-ga-mpich2  auto-build GA with MPI and MPICH2 (experimental)
-auto-ga-mvapich2 auto-build GA with MPI and MVAPICH2 (experimental)
-auto-ga-openmpi auto-build GA with MPI and Open MPI (experimental)
-auto-ga-tcgmsg  auto-build GA with TCGMSG (experimental)
-auto-ga-tcgmsg-mpich2 auto-build GA with TCGMSG-MPI and MPICH2 (experimental)
-auto-ga-tcgmsg-mvapich2 auto-build GA with TCGMSG-MPI and MVAPICH2 (experimental)
-auto-ga-tcgmsg-openmpi auto-build GA with TCGMSG-MPI and Open MPI (experimental)
-auto-mpich2   auto-build MPICH2 (experimental)
-auto-mvapich2 auto-build MVAPICH2 (experimental)
-auto-openmpi  auto-build Open MPI (experimental)
-batch        run script non-interactively
-blas         use external BLAS library
-blaspath     specify blas library path
-cc           use C compiler named cc
-clearspeed   compile clearspeed code
-clearspeedbase specify clearspeed base path for includes and libraries
```

-cuda	try to get settings for compiling CUDA code
-f	specify Molpro fortran pre-processor flags
-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
-hdf5	use external HDF5 library
-hdf5path	specify HDF5 library path
-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
-lapack	use external LAPACK library
-lapackpath	specify LAPACK library path
-letter	specify letter latex paper size
-Linux	use settings for Linux kernel
-mpp	produce parallel Molpro
-mppbase	specify mpp base path for includes and libraries
-mppx	produce trivial parallel Molpro
-nagfor	use NAG Fortran compiler
-natom	max number of atoms
-nbasis	max number of basis functions
-noblas	Don't use external BLAS library
-noclearspeed	do not compile clearspeed code
-nocuda	don't compile CUDA code
-nocxx	do not compile C++ code
-nohdf5	don't use external HDF5 library
-nolapack	don't use external LAPACK library
-nolargefiles	Do not use largefiles
-noopenmp	compile without openmp

-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
-openc	use Open64 C compiler
-openf90	use Open64 Fortran compiler
-openmp	compile with openmp
-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
-rpm	Build with RPM settings
-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
-trial	use trial parse object
-x86_64	use settings for 64-bit x86 machine
-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. It may be helpful in such circumstances to compare the target platform with the lists of platforms on which MOLPRO is thought to function at <http://www.molpro.net/supported>. 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 Getting and applying patches

Normally, the distribution when downloaded is fully up to date, and initial patching is not necessary. However, bug fixes and updates may be desired subsequently. The mechanism for updating MOLPRO source code with bug fixes and new features is through the provision of self-contained patch files, which, when applied, replace or add files, and store the replaced code in order to allow later reversion to the original. Those patches that are available can be seen at <https://www.molpro.net/download/patch?version=2010.1> whilst a list of those already installed is printed when running the program. Patch files automatically outdate any targets that need rebuilding as a result of the patch; for example, relevant object files are removed. Thus, after all patches have been applied, it is usually necessary to rebuild the program using `make`.

The order in which patches are applied and removed is important. Some patches are prerequisites of others, and some patches are ‘parents’ of one or more ‘children’: the parent and child patches have one or more files in common, but the parent is older than the child. Individual patch scripts will themselves refuse to apply or revert if rules based on these considerations would be violated. In order to deal with this issue smoothly, a program `patcher` is provided to manage the application and removal of one or more patches. `patcher` attempts to sort the order in which patches are applied or reverted so as to avoid such conflicts; it will also, if necessary, revert and reapply patches.

In order to run `patcher` you should issue the command:

```
make patch
```

This should be sufficient for most purposes. `patcher` will be built if has not yet been compiled and then it will contact the webserver, apply any available patches and then return the patchlevel that you have reached.

If it is necessary to pass arguments to the `patcher` program then in the top-level directory issue the command

```
./patcher [--apply | --revert | --list]
          [--cache-directory] [--user] [--password]
          [--url] [--local]
          [--verbose] [--no-action] patch1 patch2 ....
```

It can operate in one of three possible modes according to the options

<code>--apply, -a</code>	(default) Apply (i.e. install) patches
<code>--revert, -r</code>	Revert (i.e. remove) patches
<code>--list, -l</code>	List available and installed patches

The list of patches to remove or install can be given on the command line after all options as an explicit list of either patch names or, in the case of application, patch files. Alternatively and usually, for the case of application, one can through options request either all patches that are in a local cache, or all patches that are available.

The MOLPRO patches from the central web server (default `http://www.molpro.net`), are cached by this program in a local directory (default `$HOME/.molpro/cache`). Access to the web server typically has to be authenticated; the first time you run this program, you can specify your username and password through command-line options, or else the program will prompt for them. They are then remembered in the file `CONFIG` in the cache directory.

In case of problems, first consult the file `patcher.log`, which contains the output from individual patch applications and reversions.

The following options can be given.

<code>--cache-directory, -c d</code>	location of cache directory.
<code>--verbose, -v</code>	Increase amount of information printed. Multiple <code>--verbose</code> options can be used.
<code>--noverbose</code>	Decrease amount of information printed.
<code>--url</code>	URL of web server.
<code>--user, -u u</code>	Username for web server.
<code>--password, -p p</code>	Password for web server.
<code>--noaction, -n</code>	No applications or reversions are actually done. Useful for seeing what would happen without doing it.
<code>--local</code>	Don't attempt to access the web server, but use only local files.
<code>--token, -k</code>	Download your licence key
<code>--ssl, -s</code>	Use SSL when contacting the webserver
<code>--nossl, -i</code>	Turn off SSL use

Examples:

```
patcher
```

Applies all patches that are available, but not yet installed. This is the normal use of the utility in bringing the copy of the source tree up to date with all available updates.

```
patcher -l
```

Lists installed and available patches.

```
patcher -r xx yy
```

Reverts patches `xx` and `yy`.

```
patcher -n
```

Loads all uninstalled patches into the cache for later use.

```
patcher --local
```

Applies all patches in the cache; no network connection needed.

### 3.10 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 `molpro2010.1/doc/manual.pdf`, with the HTML version in the directory `molpro2010.1/doc/manual` (top level file is `frames.html`). The documentation can be copied to its final destination as specified in the `CONFIG` file generated by the `configure` command. To install the documentation and interactive basis set tool, issue `make install` in the `doc` directory. Numerous example input files are included in the manual, and can alternatively be seen in the directory `molpro2010.1/examples`.

### 3.11 Fedora 13 (32-bit)

Installing MOLPRO from source on Fedora 13 Linux can be done very easily using only free software. These notes assume a standard installation of Fedora 13. For all builds one should install the following additional packages via yum:

<code>gcc-c++</code>	provides GNU C and C++ compiler,
<code>gcc-gfortran</code>	provides GNU Fortran compiler,
<code>zlib-devel</code>	provides a library for zipped files.

Optionally one can choose to install:

<code>blas-devel</code>	provides a BLAS library,
<code>lapack-devel</code>	provides a LAPACK library,

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

#### 3.11.1 Serial MOLPRO

After unpacking MOLPRO one can simply type

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

to build MOLPRO.

#### 3.11.2 Parallel MOLPRO using the Global Arrays with MPI2

The following additional packages should be installed via apt-get:

<code>mpich2</code>	provides MPI-2 library files
<code>mpich2-devel</code>	provides MPI-2 include files

Set up mpd by running:

```
mpd
```

and following the on screen instructions.

The latest stable version of the Global Arrays toolkit should be downloaded. Build and test the Global Arrays toolkit with:

```
make TARGET=LINUX FC=gfortran CC=gcc USE_MPI=1 \  
    MPI_LIB=/usr/lib/mpich2/lib \  
    MPI_INCLUDE=/usr/include/mpich2-i386 LIBMPI=-lmpich  
mpirun ./global/testing/test.x
```

Then configure and build MOLPRO with:

```
./configure -batch -gcc -gfortran \  
            -mpp -mppbase /path/to/directory/ga-4-3-3  
make
```

### 3.12 openSUSE 11.3 (32 & 64-bit)

Installing MOLPRO from source on openSUSE 11.3 Linux can be done very easily using only free software. These notes assume a standard installation of openSUSE 11.3. For all builds one should install the following additional packages via YaST:

gcc-c++	provides GNU C and C++ compiler,
gcc-fortran	provides GNU Fortran compiler,
make	provides GNU make
zlib-devel	provides a library for zipped files.

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.

#### 3.12.1 Serial MOLPRO

After unpacking MOLPRO one can simply type

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

to build MOLPRO.

**3.12.2 Parallel MOLPRO using the Global Arrays with MPI**

The following additional packages should be installed via YaST:

```
mpich                provides MPI-1 library
mpich-devel          provides MPI-1 library
```

The latest stable version of the Global Arrays toolkit should be downloaded. Build and test the Global Arrays toolkit for 64-bit machines with:

```
make TARGET=LINUX64 FC=gfortran CC=gcc USE_MPI=1 \
    MPI_LIB=/opt/mpich/ch-p4/lib64 \
    MPI_INCLUDE=/opt/mpich/ch-p4/include LIBMPI=-lmpich
/opt/mpich/ch-p4/bin/mpirun ./global/testing/test.x
```

or on 32-bit machines with:

```
make TARGET=LINUX FC=gfortran CC=gcc USE_MPI=1 \
    MPI_LIB=/opt/mpich/ch-p4/lib \
    MPI_INCLUDE=/opt/mpich/ch-p4/include LIBMPI=-lmpich
/opt/mpich/ch-p4/bin/mpirun ./global/testing/test.x
```

Then configure and build MOLPRO with:

```
./configure -batch -gcc -gfortran \
            -mpp -mppbase /path/to/directory/ga-4-3-3
make
```

**3.13 Ubuntu 10.04 (32-bit)**

Installing MOLPRO from source on Ubuntu 10.04 Linux can be done very easily using only free software. These notes assume a standard installation of Ubuntu 10.04. For all builds one should install the following additional packages via apt-get:

```
build-essential      provides GNU C++ compiler,
gfortran              provides GNU Fortran compiler,
curl                  provides curl for downloading patches
zlib1g-dev            provides a library for zipped files.
```

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.

### 3.13.1 Serial MOLPRO

After unpacking MOLPRO one can simply type

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

to build MOLPRO.

### 3.13.2 Parallel MOLPRO using the Global Arrays with MPI

The following additional packages should be installed via apt-get:

mpich-bin	provides MPI-1 library
libmpich1.0-dev	provides MPI-1 library
openssh-server	provides ssh access to localhost

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.

The latest stable version of the Global Arrays toolkit should be downloaded. Build and test the Global Arrays toolkit with:

```
make TARGET=LINUX FC=gfortran CC=gcc USE_MPI=1 \
    MPI_LIB=/usr/lib/mpich/lib \
    MPI_INCLUDE=/usr/lib/mpich/include LIBMPI=-lmpich
mpirun ./global/testing/test.x
```

Then configure and build MOLPRO with:

```
./configure -batch -gcc -gfortran \
    -mpp -mppbase /path/to/directory/ga-4-3-3
make
```

## 3.14 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 contains a list of potentially useful packages.

With the above steps Molpro should now configure and will treat Cygwin as a generic Linux system. Once the program has been built it can be run in the normal way under Cygwin.

Package	Package Group	Reason
gcc4	Devel	For compiling C files
gcc4-fortran	Devel	For compiling Fortran files
gcc4-g++	Devel	For compiling C++ files
make	Devel	need GNU make
curl	Web	curl needed for patcher

Table 1: Cygwin requirements for user install

Package	Package Group	Reason
bison	Devel	bison
gdb	Devel	gdb
libxslt	Gnome	xsltproc
openssh	Net	ssh
vim	Editors	vi
wget	Web	wget, alternative to curl for patcher

Table 2: Cygwin packages for developers

### 3.15 Parallel installation on 64-bit MacOS 10.6.4 standalone machine

This section contains notes and examples for building some of the programs needed for building parallel Molpro. It is not complete or extensive, and in all cases one should refer directly to the documentation of the program in question and forward any queries directly there.

#### 3.15.1 Building MPICH2-1.3

The following instructions are for building MPICH2 with ifort and installing in `/usr/local/mpich2-nemesis`.

```
./configure --with-device=ch3:nemesis --with-pm=gforker \
--prefix=/usr/local/mpich2-nemesis --enable-fc FC=ifort \
--enable-cc CC=gcc --enable-cxx CXX=g++
```

```
make
sudo make install
```

#### 3.15.2 Building Global Arrays

The following instructions are for building GA using the MPICH2 installation detailed in the previous section.

```
make CC=gcc FC=ifort TARGET=MACX64 USE_MPI=y \
MPI_LIB=/usr/local/mpich2-nemesis/lib \
MPI_INCLUDE=/usr/local/mpich2-nemesis/include \
LIBMPI="-lmpich -lmpich -lopa -lmpi"
```

```
export PATH=/usr/local/mpich2-nemesis/bin:$PATH
mpiexec -n 2 global/testing/test.x
```

Note that if `___emutls_get_address` is reported as an undefined reference, then `-lgcc_eh` must be appended to `LIBMPI`.