PETSc

- Home

Docs: Installation

Quick instructions:

Invoke the following commands from the top level PETSc directory [using bash shell]

```
export PETSC_DIR=$PWD
./config/configure.py --with-cc=gcc --with-fc=gfortran --download-f-blas-lapack=1 --download-mpich=1
make all test
```

Detailed instructions: [Check here for MS Windows Installation]

- Sugest downloading and installing PETSc as a **regular/non-root user,** perhaps in **/home/username/soft**
- Download latest PETSc release tarball: petsc-3.1-p0.tar.gz
- `cd /home/username/soft`
- `gunzip -c petsc-3.1-p0.tar.gz | tar -xof -`
- `cd petsc-3.1-p0`
- sh/bash shell: `PETSC_DIR=$PWD; export PETSC_DIR`
  csh/tcsh shell: `setenv PETSC_DIR $PWD`
- `./config/configure.py` (use --help for options or the example usages below)
- `make all`
- `make test`

Encounter problems?

- **Read the error message from ./config/configure.py!**
- If you get the message 'No such file or directory' try `python config/configure.py`
- BLAS and LAPACK problems
- MPI problems. I Don't want MPI
- Trouble using other external packages
- Check the bug-reporting section

Example Usages:

- Examples are at **config/examples/*.py**. We use some of these scripts locally for testing - for example one can update these files and run as:
  ```
  config/examples/arch-osx-10.6.py
  ```
- Using the bash shell, assuming BLAS, LAPACK, MPICH are not currently installed ./config/configure.py will download & install BLAS, LAPACK, MPICH if they are not already installed on the system) :
  ```
  export PETSC_DIR=/home/petsc/petsc-3.1-p0
  cd $PETSC_DIR
  ./config/configure.py --with-cc=gcc --with-fc=gfortran --download-f-blas-lapack=1 --download-mpich=1
  make
  make test
  ```
- Same as above - but build Complex version of PETSc [using c++ compiler] (add the option --with-fortran-kernels=generic to get possibly faster complex number performance on some systems):
  ```
  export PETSC_DIR=/home/petsc/petsc-3.1-p0
  cd $PETSC_DIR
  ./config/configure.py --with-cc=gcc --with-fc=gfortran --with-cxx=g++ --download-f-blas-lapack=1 --download-mpich=1 --with-scalar-type=complex --with-clanguage=cxx
  make
  make test
  ```
- Using the bash shell, assuming BLAS, LAPACK, MPICH software are installed at the specified locations, and use MPI compilers **mpicc/mpif90**:
  [Note: Do not specify --with-cc --with-fc etc when using --with-mpi-dir - so that mpicc/mpif90 can be picked up from mpi-dir]
  ```
  export PETSC_DIR=/home/petsc/petsc-3.1-p0
  cd $PETSC_DIR
  ./config/configure.py --with-blas-lapack-dir=/usr/local/lib --with-mpi-dir=/usr/local/mpich
  make
  make test
  ```
- Using csh shell, install 2 variants of PETSc, one with gnu, the other with intel compilers
  ```
  ./config/configure.py PETSC_ARCH=linux-gnu CC=gcc FC=gfortran --download-mpich=1
  make PETSC_ARCH=linux-gnu
  make PETSC_ARCH=linux-gnu test
  ./config/configure.py PETSC_ARCH=linux-gnu-intel CC=icc FC=ifort --download-mpich=1 --with-blas-lapack-dir=/usr/local/mkl
  make PETSC_ARCH=linux-gnu-intel
  make PETSC_ARCH=linux-gnu-intel test
  ```

PETSC_DIR and PETSC_ARCH are a couple of variables control the configuration and build process of PETSc. These variables can be set as envirnment variables or specified on the command line [to both configure and make]

- specify enviornment variable for csh/tcsh [can be specified in ~/.cshrc]

  ```
  setenv PETSC_DIR /home/balay/petsc-3.1-p0
  setenv PETSC_ARCH linux-gnu-c-debug
  ```

- specify enviornment variable for bash [can be specified in ~/.bashrc]

```
export PETSC_DIR=/home/balay/petsc-3.1-p0
export PETSC_ARCH=linux-gnu-c-debug
```

- specify variable on commandline to configure

```
config/configure.py PETSC_DIR=/home/balay/petsc-3.1-p0 PETSC_ARCH=linux-gnu-c-debug [other configure options]
```

- specify variables on command line to make

```
make PETSC_DIR=/home/balay/petsc-3.1-p0 PETSC_ARCH=linux-gnu-c-debug [other make options]
```

PETSC_DIR: this variable should point to the location of the PETSc installation that is used. Multiple PETSc versions can coexist on the same file-system. By changing PETSC_DIR value, one can switch between these installed versions of PETSc.

PETSC_ARCH: this variable gives a name to a configuration/build. Configure uses this value to stores the generated config makefiles in ${PETSC_DIR}/${PETSC_ARCH}/conf. And make uses this value to determine this location of these makefiles [which intern help in locating the correct include and library files].

Thus one can install multiple variants of PETSc libraries - by providing different PETSC_ARCH values to each configure build. Then one can switch between using these variants of libraries [from make] by switching the PETSC_ARCH value used.

If configure doesn't find a PETSC_ARCH value [either in env variable or command line option], it automatically generates a default value and uses it. Also - if make doesn't find a PETSC_ARCH env variable - it defaults to the value used by last successful invocation of previous configure.

Return to Installation Instructions

Compilers: Specify compilers and compiler options used to build PETSc [and perhaps external packages]

- Specify compilers using the options --with-cc --with-fc --with-cxx for c, c++, fortran compilers
  - --with-cc=mpicc --with-fc=mpif90
  - --with-cc=gcc --with-fc=gfortran
  - --with-cc=gcc --with-cxx=g++ --with-fc=gfortran --with-clanguage=cxx
- If fortran compiler is not available - then disabling using fortran
  - --with-fc=0
- If no compilers are specified - configure will automatically look for available MPI or regular compilers in users PATH
  - mpicc/mpiCC/mpif90 or mpif77
  - gcc/g++/gfortran or g77
  - cc/CC/f77 etc..
- Its best to use MPI compilers as this will avoid the situation where MPI is compiled with one set of compilers [like gcc/g77] and user specified incompatible compilers to PETSc [perhaps icc/ifort]. This can be done by either specifying --with-cc=mpicc or --with-mpi-dir [and not --with-cc=gcc]
  - --with-cc=mpicc --with-fc=mpif90
  - --with-mpi-dir=/opt/mpich2-1.1 [but *no* --with-cc=gcc]
- Configure defaults to building PETSc in debug mode. One can switch to using optimzed mode with the toggle option --with-debugging [defaults to debug enabled]. Additionally one can specify more suitable optimization flags with the options COPTFLAGS, FOPTFLAGS, CXXOPTFLAGS.
  - ./config/configure.py --with-cc=gcc --with-fc=gfortran --with-debugging=0 COPTFLAGS='-O3 -march=p4 -mtune=p4' FOPTFLAGS='-O3 -qarch=p4 -qtune=p4'

- Configure cannot detect compiler libraries for certain set of compilers. In this case one can specify additional system/compiler libraries using the LIBS option
  - ./config/configure.py LIBS='-ldl /usr/lib/libm.a'

Return to Installation Instructions

External Packages: PETSc provides interfaces to various external packages. Blas/Lapack and MPI are generally required packages - but one can optionally use external solvers like Hypre, MUMPS etc.. from within PETSc aplications.

PETSc configure has the ability to download and install these external packages. Alternatively if these packages are already installed, then configure can detect and use them.

The following modes can be used to install/use external packages with configure.

--download-PACKAGENAME=1 : Download specified package and install it. Then configure PETSc to use this package.

- --download-f-blas-lapack=1 --download-mpich=1
- --download-blacs=1 --download-scalapack=1 --download-mumps=1

--download-PACKAGENAME=/PATH/TO/package.tar.gz : If config/configure.py cannot automatically download the package [due to network/firewall issues], one can download the package by alternaive means [perhaps wget or scp via some other machine]. Once the tarfile is downloaded, the path to this file can be specified to configure with this option. Configure will proceed to install this package and then configure PETSc with it.

- --download-mpich=/home/petsc/mpich2-1.0.4p1.tar.gz

--with-PACKAGENAME-dir=PATH : If the external package is already installed - specify its location to configure [it will attempt to detect, include, library files from this location.] Normally this corresponds to the top-level installation dir for the package.

- --with-mpi-dir=/home/petsc/software/mpich2-1.0.4p1

--with-PACKAGENAME-include=INCLUDEPATH --with-PACKAGENAME-lib=LIBRARYLIST: Usually a package is defined completely by its include file location - and library list. [If the package is already installed] - then one can use these two options to specify the package to configure.

- --with-parmetis-include=/home/petsc/software/parmetis/include --with-parmetis-lib=/home/petsc/software/parmetis/lib/libparmetis.a
- --with-mpi-include=/home/petsc/software/mpich2-1.0.4p1/include --with-mpi-lib=[/home/petsc/software/mpich2-1.0.4p1/lib/libmpich.a,-lpthread,-lrt]

Notes:

- Run **config/configure.py-help** to get the list of external packages - and corresponding additional options [for example --with-mpiexec for mpich]
- Generally one would use either one of the above 4 modes for any given package - and not mix these. [i.e mixing -with-mpi-dir and -with-mpi-include etc.. should be avoided]
- Some packages might not support certain options like --download-PACKAGENAME or --with-PACKAGENAME-dir. Some architectures like Windows might have issues with these options. In these cases, --with-PACKAGENAME-include --with-PACKAGENAME-lib options should be prefered.
- Its best to install some external packages like SuperLU_DIST, MUMPS, Hypre with the option --download-PACKAGENAME. [the correct options for these packages are --download-superlu_dist=1 --download-mumps=1 --download-hypre=1]
  - This will install the COMPATIBLE version of the external package. A generic install of this package might not be compatible with PETSc [perhaps due to version

differences - or perhaps due to the requirement of additional patches for it to work with PETSc]
  - All packages will be installed with the same set of compilers - this avoids problems [for ex: wiered link time errors] with mixing code compiled with multiple compilers [for example mixing gfortran and ifort compiled code].
- If one had to download a compatible external package manually, then the URL for this package is listed in configure source for this package. For example, check config/PETSc /packages/SuperLU.py for the url for download this package.

Additional options:

--with-external-packages-dir=PATH : By default, external packages will be installed in ${PETSC_DIR}/externalpackages. However one can choose a different location where these packages are installed.

Return to Installation Instructions

BLAS/LAPACK: these packages provide some basic numeric kernels used by PETSc.

- Configure will automatically look for blas/lapack in certain standard locations, on most systems you should not need to provide any information about BLAS/LAPACK in the config/configure.py command, some of the standard locations used are
  - /usr/lib/libblas.a,liblapack.a
  - Intel MKL on Windows or Linux
  - sunperf on solaris
  - VecLib on Macs
  - IBM ESSL
  - [and many more]
- One can use the following options to let configure download/install blas automatically.
  - --download-f-blas-lapack=1 [when fortran compiler is present]
  - --download-c-blas-lapack=1 [when configuring without a fortran compiler - i.e --with-fc=0]
- Alternatively one can use other externalpackages installation options like the following.
  - --with-blas-lapack-lib=libsunperf.a
  - --with-blas-lib=libblas.a --with-lapack-lib=liblapack.a
  - --with-blas-lapack-dir=opt/intel/mkl72

Notes:

- Sadly, IBM's ESSL does not have all the private routines of BLAS that some packages, such as SuperLU expect; in particular slamch, dlamch and xerbla. Therefor you need a full implementation of blas/lapack when using these package. In this case, use --download-f-blas-lapack=yes

Return to Installation Instructions

MPI: This software provides the parallel functionality for PETSc.

- Configure will automatically look for MPI compilers mpicc/mpif77 etc and use them if found [in default PATH]
- One can use the following options to download/install

- One can use the following options to let configure download/install MPI automatically
    - --download-mpich=1 [install and use MPICH]
    - --download-openmpi=1 [Install and useOpenMPI]
- Alternatively one can use other externalpackages installation options.

Using MPI Compilers:

- Its best to install PETSc with MPI compilers - this way, the SAME compilers used to build MPI are used to build PETSc [this avoids incompatibilities which might crop up - when using libraries compiled with different c or fortran compilers.]. This can be achieved with the following modes.
    - Vendor provided MPI might already be installed. IBM, SGI, Cray etc provide their own.
      ./config/confiure.py --with-cc=mpcc --with-fc=mpf77
    - If using MPICH which is already installed [perhaps using myrinet/gm] then use [without specifying --with-cc=gcc etc.so that configure picks up mpicc from mpi-dir]:
      ./config/configure.py --with-mpi-dir=/path-to-mpich-install

Installing without MPI:

- You can build (sequential) PETSc without an MPI. This is useful for quickly installing PETSc [if MPI is not available - for whatever reason]. However - if there is any MPI code in user application, then its best to install a full MPI - even if the usage is currently limited to uniprocessor mode.
    - ./config/configure.py --with-mpi=0

Installing with OpenMPI with shared MPI libraries:

OpenMPI defaults to building shared libraries for MPI. However, the binaries generated by MPI wrappers mpicc/mpif77 etc require LD_LIBRARY_PATH to be set to the location of these libraries.

Due to this OpenMPI restriction one has to set LD_LIBRARY_PATH correctly [per OpenMPI installation instructions], before running PETSc configure. The same issue might exist with LAM as well.

Notes:

- Avoid specifing compilers [with options --with-cc or --with-fc] when using the option --with-mpi-dir. [Option--with-mpi-dir specifies using MPI compilers - so its best to use them - and not overwride them with user specified --with-cc].
- One can specify mpiexec or mpiexec with the options --with-mpiexec

MPI	http://www.mpi-forum.org
MPICH	http://www.mcs.anl.gov/mpi/mpich
OpenMPI	http://www.open-mpi.org

Return to Installation Instructions

Microsoft Windows Installation With MS/Intel/Compaq Compilers:

Microsoft windows OS does not provide the same unix shell enviornment as the other OSes. Also the default MS/Intel/Compaq compilers behave differently than other unix compilers. And PETSc primarily relies on a unix envirment for build tools. So to install PETSc on windows - one has to install cygwin [for the unix enviornment] and use win32fe [part of PETSc sources,to interface to MS/Intel/Compaq compilers].

Install Cygwin: Please download and install cygwin package from http://www.cygwin.com Make sure the following cygwin components are installed.

- python
- make
- [default selection should already have diff and other tools]

Remove Cygwin link.exe: Cygwin link.exe can conflict with Intel ifort and Comapq F90 compilers. If you are using these compilers - please do [from cygwin/bash shell]:

- `mv /usr/bin/link.exe /usr/bin/link-cygwin.exe`

Setup cygwin bash shell with Working Compilers: We require the compilers to be setup properly in a cygwin bash command shell, so that "`cl foo.c`" or "`ifort foo.f`" works from this shell. For example - if using VS2005 C and Intel 10 Fortran one can do:

- Start -> Programs -> Intel Software Development Tools -> Intel Fortran Compiler 10 -> Visual Fortran Build Enviornment [32bit or 64bit depending on your usage]. This should start a 'dos cmd' shell.
- Within this shell - run cygwin bash.exe as: `c:\cygwin\bin\bash.exe --login`
- verify if the compilers are useable [by running cl, ifort in this shell]
- Now run configure with win32fe and then build the libraries with make [as per the usual instructions]

Notes: This a new requirement as win32fe can no longer autodetect the newer versions of compilers. They currently default to "noautodetect mode.

Example Configure usage with Windows Compilers:

Use configure with VC2005 C and Intel Fortran 10 [With MPICH2 installed].

```
./config/configure.py --with-cc='win32fe cl' --with-fc='win32fe ifort'  --with-cxx='win32fe cl'  --download-f-blas-lapack=1
```

If fortran usage is not required, use:

```
./config/configure.py --with-cc='win32fe cl' --with-fc=0 --download-c-blas-lapack=1
```

Using Compaq F90:

Using Microsoft C/C++ 6.0 & Compaq Fortran 6.0 with MPICH2 configure command to use:

```
./config/configure.py --with-cc='win32fe cl' --with-fc='win32fe f90' --download-f-blas-lapack=1
```

Note: MPICH2 mpif.h needs a fix for it to work with Compaq F90 [specifically remove line with MPI_DISPLACEMENT_CURRENT - which uses 'integer*8' - which is unsupported by Compaq F90]

ExternalPackages: --download-package option does not work with many external packages.

Project Files: We provide templates for Microsoft Visual Studio project files at ${PETSC_DIR}/projects. These work for us - with our configure build [config/cygwin-ms.py with MPICH1 and Intel MKL 5], However they will REQURE MODIFICATIONS to work with a user build of PETSc - as the locations of packages & configure options used by user could be different from our defult build.

To get these project files working for your installation of PETSc, please do the following:

- try compiling the example from cygwin shell - using makefile - for eg:
  cd src/ksp/ksp/examples/tutorials
  make ex2
- if the above works - then make sure all the compiler/linker options used by make are also present in the project file in the correct notation.
- if errors - redo the above step. [if all the options are correctly specified - then the example should compile from MSDev.

Debugger: Running PETSc probrams with -start_in_debugger is not supported on this platform, so debuggers will need to be initiated manually. Make sure your environment is properly configured to use the appropriate debugger for your compiler. The debuggers can be initiated using Microsoft Visual Studio 6: **msdev ex1.exe**, Microsoft Visual Studio .NET: **devenv ex1.exe**, Intel Enhanced Debugger: **edb ex1.exe**, or GNU Debugger **gdb ex1.exe**.

Using Cygwin gcc/g++/gfortran: One can install and use PETSc with gcc/gfortran compilers from cygwin. In this case follow the regular instructions.

PETSc Win32 front end - win32fe: This tool is used as a wrapper to Microsoft/ Borland/ Intel compilers and associated tools - to enable building PETSc libraries using cygwin make and other UNIX tools. For additional info, run ${PETSC_DIR}/bin/win32/win32fe without any options.

Return to Installation Instructions

Installing PETSc in /usr/local or /opt where sudo or root previledges are required:

If one wants to install PETSc [with sources] in a common system location like /usr/local or /opt, then sugest creating a dir for PETSc in the required location with user previledges, and then do the PETSc install [as a **regular/non-root user**]. i.e

- sudo mkdir /opt/petsc
- sudo chown user:group /opt/petsc
- cd /opt/petsc
- tar -xzf petsc-3.1-p0.tar.gz
- cd petsc-3.1-p0
- ./config/configure.py
- make

One can also use the gnu prefix-install mode.

- [untar PETSc in a non-root regular location - say /home/username]
- setenv PETSC_DIR $PWD
- ./config/configure.py --prefix=/opt/petsc/petsc-3.1-p0 [other configure options]
- make
- sudo make install PETSC_DIR=$PWD

After the install is done, one has to switch to using PETSC_DIR=/opt/petsc/petsc-3.1-p0. If you've installed PETSc with the --prefix option then you DO NOT use a PETSC_ARCH variable. You should install different configurations using different --prefix names.

Installing on machine requiring cross compiler or a job scheduler:

If one has to use a cross compiler - or go through the job scheduler to use MPI on a given machine - use the configure option **'--with-batch=1'** as follows:

- run configure with the additional option **'--with-batch=1'** on the **frontend node** (compiler server) [perhaps with the additional option **'--known-mpi-shared=0']**
- the above configure run will create a binary **'conftest'**. Run this binary 'conftest' on **one compute node** using the job scheduler.
- The above run of conftest will create a new python script **'reconfigure'**. Run **'python reconfigure'** again on the **frontend node** (compiler server)to complete the configuration process

Installing on IBM SP:

To run any code compiled with MPI on IBM SP - one has to go through the **job scheduler**. [So follow the instructions from the above section 'Installing on machine requiring job scheduler']. Alternatively check/run config/examples/aix5.1.0.0.py for all configure options required on the SP with compilers defaulting to 32bit mode, and config/examples /aix5.1.0.0-64.py with compilers defaulting to 64bit mode.

Installing on IBM BG/L:

IBM BG/L [as of now] requires modified compiler script wrappers to compile PETSc-3.1-p0. Please download http://ftp.mcs.anl.gov/pub/petsc/tmp/petsc-bgl-tools.tar.gz and follow instructions provided in the **README** file

Installing on IBM BG/P:

IBM BG/P has mpixlc and mpixlf wrappers so no additional compiler wrappers are needed.

Installing with TAU Instrumentation package:

TAU package and the prerequisite PDT packages need to be installed separately. PETSc provides a wrapper compiler for TAU. This needs to be invoked with the correct information for the installed TAU and PDT packages. For eg:

```
./config/configure.py --with-mpi-dir=/home/petsc/soft/linux-rh73/mpich-1.2.4 --with-fc=0 --with-cxx=0 -PETSC_ARCH=linux-tau --with-cc="`pwd`/bin/taucc.py -cc=gcc
-pdt_parse=/homes/petsc/soft/linux-rh73/pdtoolkit-2.2b1/linux/bin/cparse -tau_lib_dir=/homes/petsc/soft/linux-rh73/tau-2.11.18/i386_linux/lib"
```

Installing TOPS components:

To install TOPS components one has to download and install CCA Tools. [linux only]

- Get the CCA Tools package from http://www.cca-forum.org/download/cca-tools/cca-tools-0.5.9.tar.gz
- Configure cca-tools using '-with-mpi' option [prerequisites might be Java SDK, c++, mpich1]
- Configure PETSc with C++, babel, ccafe options. Check config/config/asterix-tops.py for an example.
- Build PETSc libraries & TOPS components with 'make all'
- To test do: 'cd src/tops; make examples'

Return to Installation Instructions
