**If you use an Intel compiler, you must also make sure that you have a version of GCC that supports C++11 (GCC 4.8+). You also need to set the environment variables properly, like: export export PATH=${HOME}/opt-gcc/gcc/bin:$PATH**

**Time: 20190320**

**Environment 1**： System：deepin15.9.2

Compiler: gcc version 8.1.0 (GCC)

**Install third-party codes:**

See appendix A.

**Set environment varibles:**

#========== gcc environment variables ================

export PATH=${HOME}/opt-gcc/gcc/bin:$PATH

export LD\_LIBRARY\_PATH=${HOME}/opt-gcc/gcc/lib64:$LD\_LIBRARY\_PATH

export LD\_LIBRARY\_PATH=${HOME}/opt-gcc/gmp/lib:$LD\_LIBRARY\_PATH

export LD\_LIBRARY\_PATH=${HOME}/opt-gcc/mpc/lib:$LD\_LIBRARY\_PATH

export LD\_LIBRARY\_PATH=${HOME}/opt-gcc/mpfr/lib:$LD\_LIBRARY\_PATH

#========== BOUT++ environment variables ================

export third\_part\_codes\_path=${HOME}/opt-gcc

export bout\_home\_dir=${HOME}/codes/BOUT-dev-4.2.0

export gcc\_dir=${third\_part\_codes\_path}/gcc

export mpich\_dir=${third\_part\_codes\_path}/mpich

export hdf5\_dir=${third\_part\_codes\_path}/hdf5

export netcdf\_dir=${third\_part\_codes\_path}/netcdf

export fftw\_dir=${third\_part\_codes\_path}/fftw

export sundials\_dir=${third\_part\_codes\_path}/sundials

export PATH=${gcc\_dir}/bin:$PATH

export PATH=${mpich\_dir}/bin:$PATH

export PATH=${fftw\_dir}/bin:$PATH

export PATH=${netcdf\_dir}/bin:$PATH

export LD\_LIBRARY\_PATH=${gcc\_dir}/lib64:$LD\_LIBRARY\_PATH

export LD\_LIBRARY\_PATH=${mpich\_dir}/lib:$LD\_LIBRARY\_PATH

export LD\_LIBRARY\_PATH=${hdf5\_dir}/lib:$LD\_LIBRARY\_PATH

export LD\_LIBRARY\_PATH=${netcdf\_dir}/lib:$LD\_LIBRARY\_PATH

export LD\_LIBRARY\_PATH=${fftw\_dir}/lib:$LD\_LIBRARY\_PATH

export LD\_LIBRARY\_PATH=${sundials\_dir}/lib:$LD\_LIBRARY\_PATH

#========= python3 ==================

export anaconda3\_dir=${third\_part\_codes\_path}/anaconda3

export PATH=${anaconda3\_dir}/bin:$PATH

export PYTHONPATH=${bout\_home\_dir}/tools/pylib/:$PYTHONPATH

export HDF5\_DISABLE\_VERSION\_CHECK=1

**Configure options:**

./configure --with-fftw=${fftw\_dir} --with-hdf5=${hdf5\_dir}/bin/h5cc --with-netcdf=${netcdf\_dir} --with-sundials=${sundials\_dir}

**Appendix A**

:' for centos 6, set the following environments in the file ~/.bashrc

export PATH=/home/wphu/opt-gcc/gcc/bin:$PATH

export LD\_LIBRARY\_PATH=${HOME}/opt-gcc/gcc/lib64:$LD\_LIBRARY\_PATH

export LD\_LIBRARY\_PATH=${HOME}/opt-gcc/gmp/lib:$LD\_LIBRARY\_PATH

export LD\_LIBRARY\_PATH=${HOME}/opt-gcc/mpc/lib:$LD\_LIBRARY\_PATH

export LD\_LIBRARY\_PATH=${HOME}/opt-gcc/mpfr/lib:$LD\_LIBRARY\_PATH

before installing netcdf, first install: sudo apt install curl libcurl4-openssl-dev

'

#:' for gcc compilers

export install\_path\_header=${HOME}/opt-gcc

export compiler\_c=gcc

export compiler\_cxx=g++

export compiler\_fortran=gfortran

export compiler\_mpicc=mpicc

export compiler\_mpicxx=mpicxx

export source\_codes\_root\_path=$(pwd)

export compile\_cores\_number=10

#'

:' for intel compilers

source ${HOME}/opt-intel/intel/bin/compilervars.sh intel64

export install\_path\_header=${HOME}/opt-intel

export compiler\_c=icc

export compiler\_cxx=icc

export compiler\_fortran=ifort

export compiler\_mpicc=mpicc

export compiler\_mpicxx=mpicxx

export source\_codes\_root\_path=$(pwd)

export compile\_cores\_number=10

'

# install anaconda3

bash ./Anaconda3-2018.12-Linux-x86\_64.sh -b -p ${install\_path\_header}/anaconda3

# install mpich3

export CC=${compiler\_c}

export CXX=${compiler\_cxx}

export FC=${compiler\_fortran}

package=mpich-3.2.1

install\_path=mpich

if [ -d ${install\_path\_header}/${install\_path} ];then

echo "${package} has been installed"

else

tar -xvf ${package}.tar.gz

tar -xvf ${package}.tar

tar -xvf ${package}.gz

cd ${package}

./configure --prefix=${install\_path\_header}/${install\_path}

make -j${compile\_cores\_number}

make install

cd ..

fi

export PATH=${install\_path\_header}/${install\_path}/bin:$PATH

# install hdf5

export CC=${compiler\_c}

export CXX=${compiler\_cxx}

export FC=${compiler\_fortran}

export CFLAGS=-fPIC

package=hdf5-1.8.20

install\_path=hdf5

if [ -d ${install\_path\_header}/${install\_path} ];then

echo "${package} has been installed"

else

tar -xvf ${package}.tar.gz

tar -xvf ${package}.tar

tar -xvf ${package}.gz

cd ${package}

./configure --prefix=${install\_path\_header}/${install\_path}

make -j${compile\_cores\_number}

make install

cd ..

fi

export CFLAGS=""

# install hdf5-mpich

export CC=${compiler\_mpicc}

package=hdf5-1.8.20

install\_path=hdf5-mpich

if [ -d ${install\_path\_header}/${install\_path} ];then

echo "${package} has been installed"

else

tar -xvf ${package}.tar.gz

tar -xvf ${package}.tar

tar -xvf ${package}.gz

cd ${package}

./configure --enable-parallel --prefix=${install\_path\_header}/${install\_path}

make -j${compile\_cores\_number}

make install

cd ..

fi

# install fftw

export CC=${compiler\_c}

package=fftw-3.3.4

install\_path=fftw

if [ -d ${install\_path\_header}/${install\_path} ];then

echo "${package} has been installed"

else

tar -xvf ${package}.tar.gz

tar -xvf ${package}.tar

tar -xvf ${package}.gz

cd ${package}

./configure --prefix=${install\_path\_header}/${install\_path}

make -j${compile\_cores\_number}

make install

cd ..

fi

# install netcdf(netcdf-c)

export CC=${compiler\_c}

export CPPFLAGS="-I${install\_path\_header}/hdf5/include"

export LDFLAGS="-L${install\_path\_header}/hdf5/lib"

package=netcdf-4.6.0

install\_path=netcdf

if [ -d ${install\_path\_header}/${install\_path} ];then

echo "${package} has been installed"

else

tar -xvf ${package}.tar.gz

tar -xvf ${package}.tar

tar -xvf ${package}.gz

cd ${package}

./configure --prefix=${install\_path\_header}/${install\_path}

make -j${compile\_cores\_number}

make install

cd ..

#rm -rf ${package}

fi

export CPPFLAGS=""

export LDFLAGS=""

# install netcdf-cxx

export CC=${compiler\_c}

export CXX=${compiler\_cxx}

export CPPFLAGS="-I${install\_path\_header}/hdf5/include -I${install\_path\_header}/netcdf/include"

export LDFLAGS="-L${install\_path\_header}/hdf5/lib -L${install\_path\_header}/netcdf/lib"

package=netcdf-cxx4-4.3.0

install\_path=netcdf

if [ -d ${install\_path\_header}/${install\_path} ];then

tar -xvf ${package}.tar.gz

tar -xvf ${package}.tar

tar -xvf ${package}.gz

cd ${package}

./configure --prefix=${install\_path\_header}/${install\_path}

make -j${compile\_cores\_number}

make install

cd ..

#rm -rf ${package}

fi

export CPPFLAGS=""

export LDFLAGS=""

# install netcdf-cxx

export CC=${compiler\_c}

export CPPFLAGS="-I${install\_path\_header}/hdf5/include -I${install\_path\_header}/netcdf/include"

export LDFLAGS="-L${install\_path\_header}/hdf5/lib -L${install\_path\_header}/netcdf/lib"

package=netcdf-cxx4-4.3.0

install\_path=netcdf

if [ -d ${install\_path\_header}/${install\_path} ];then

tar -xvf ${package}.tar.gz

tar -xvf ${package}.tar

tar -xvf ${package}.gz

cd ${package}

./configure --prefix=${install\_path\_header}/${install\_path}

make -j${compile\_cores\_number}

make install

cd ..

#rm -rf ${package}

fi

export CPPFLAGS=""

export LDFLAGS=""

# install lapack

export FORTRAN=${compiler\_fortran}

export FFLAGS="-fPIC"

OPTS="-O2 -frecursive"

DRVOPTS=${OPTS}

NOOPT="-O0 -frecursive"

package=lapack-3.8.0

install\_path=lapack

if [ -d ${install\_path\_header}/${install\_path} ];then

echo "${package} has been installed"

else

tar -xvf ${package}.tar.gz

tar -xvf ${package}.tar

tar -xvf ${package}.gz

cd ${package}

mkdir build

cd build

mkdir ${install\_path\_header}/${install\_path}

cmake ../ -DCMAKE\_INSTALL\_PREFIX=${install\_path\_header}/${install\_path}

make -j${compile\_cores\_number}

make install

cd ../..

fi

export FFLAGS=""

export OPTS=""

export DRVOPTS=""

export NOOPT=""

# install OpenBLAS

export FC=${compiler\_fortran}

export F77=${compiler\_fortran}

export FFLAGS=-fPIC

package=OpenBLAS-0.2.20

install\_path=OpenBLAS

if [ -d ${install\_path\_header}/${install\_path} ];then

echo "${package} has been installed"

else

tar -xvf ${package}.tar.gz

tar -xvf ${package}.tar

tar -xvf ${package}.gz

cd ${package}

mkdir build

cd build

mkdir ${install\_path\_header}/${install\_path}

cmake ../ -DCMAKE\_INSTALL\_PREFIX=${install\_path\_header}/${install\_path}

make -j${compile\_cores\_number}

make install

cd ../..

fi

export FFLAGS=""

# install SuperLU

export CC=${compiler\_c}

package=superlu\_5.2.1

install\_path=superlu

if [ -d ${install\_path\_header}/${install\_path} ];then

echo "${package} has been installed"

else

tar -xvf ${package}.tar.gz

tar -xvf ${package}.tar

tar -xvf ${package}.gz

mv SuperLU\_5.2.1 ${package}

cd ${package}

mkdir build

cd build

mkdir ${install\_path\_header}/${install\_path}

cmake ../ -DCMAKE\_INSTALL\_PREFIX=${install\_path\_header}/${install\_path}

make -j${compile\_cores\_number}

make install

cp CBLAS/libblas.a ${install\_path\_header}/${install\_path}/lib/libblas.a

cd ../..

fi

# install SuperLU-DIST

export CC=${compiler\_mpicc}

package=superlu\_dist\_5.3.0

install\_path=superlu\_dist

if [ -d ${install\_path\_header}/${install\_path} ];then

echo "${package} has been installed"

else

tar -xvf ${package}.tar.gz

tar -xvf ${package}.tar

tar -xvf ${package}.gz

mv SuperLU\_DIST\_5.3.0 ${package}

cd ${package}

mkdir build

cd build

mkdir ${install\_path\_header}/${install\_path}

cmake ../ \

-DCMAKE\_C\_FLAGS="-std=c99 -g" \

-Denable\_blaslib=OFF \

-Denable\_parmetislib=OFF \

-DBUILD\_SHARED\_LIBS=ON \

-DCMAKE\_C\_COMPILER=${compiler\_mpicc} \

-DCMAKE\_INSTALL\_PREFIX=${install\_path\_header}/${install\_path}

make -j${compile\_cores\_number}

make install

cd ../..

fi

# install PETSc

export CC=${compiler\_c}

export FC=${compiler\_fortran}

package=petsc-3.8.3

install\_path=petsc

if [ -d ${install\_path\_header}/${install\_path} ];then

echo "${package} has been installed"

else

tar -xvf ${package}.tar.gz

tar -xvf ${package}.tar

tar -xvf ${package}.gz

cd ${package}

python2 ./configure --with-mpi-dir=${install\_path\_header}/mpich --download-fblaslapack --prefix=${install\_path\_header}/${install\_path}

make -j${compile\_cores\_number} PETSC\_DIR=${source\_codes\_root\_path}/${package} PETSC\_ARCH=arch-linux2-c-debug all

make -j${compile\_cores\_number} PETSC\_DIR=${source\_codes\_root\_path}/${package} PETSC\_ARCH=arch-linux2-c-debug install

cd ..

fi

# sundials

export CC=${compiler\_c}

export FC=${compiler\_fortran}

package=sundials-2.7.0

install\_path=sundials

if [ -d ${install\_path\_header}/${install\_path} ];then

echo "${package} has been installed"

else

tar -xvf ${package}.tar.gz

tar -xvf ${package}.tar

tar -xvf ${package}.gz

cd ${package}

mkdir sundials-build

cd sundials-build

cmake \

-DCMAKE\_INSTALL\_PREFIX=${install\_path\_header}/${install\_path} \

-DEXAMPLES\_INSTALL\_PATH=${install\_path\_header}/${install\_path}/examples \

-DCMAKE\_LINKER=${install\_path\_header}/${install\_path}/lib \

-DLAPACK\_ENABLE=ON \

-DOPENMP\_ENABLE=ON \

-DMPI\_ENABLE=ON \

../

make -j${compile\_cores\_number}

make install

cd ../..

fi

# install umfpack included in SuiteSparse

export CC=${compiler\_c}

export FC=${compiler\_fortran}

export CFLAGS="-fPIC"

export CPPFLAGS="-fPIC"

package=SuiteSparse-5.3.0

install\_path=SuiteSparse

export LD\_LIBRARY\_PATH="${LD\_LIBRARY\_PATH}:${source\_codes\_root\_path}/${package}/lib"

if [ -d ${install\_path\_header}/${install\_path} ];then

echo "${package} has been installed"

else

tar -xvf ${package}.tar.gz

tar -xvf ${package}.tar

tar -xvf ${package}.gz

mv SuiteSparse ${package}

rm -rf SuiteSparse

cd ${package}

make -j${compile\_cores\_number} BLAS="${install\_path\_header}/lapack/lib/libblas.a -lgfortran" LAPACK=${install\_path\_header}/lapack/lib/liblapack.a

#make BLAS="/home/huwanpeng/source-codes/lapack-3.8.0/librefblas.a -lgfortran" LAPACK=/home/huwanpeng/source-codes/lapack-3.8.0/liblapack.a

echo $LD\_LIBRARY\_PATH

mkdir ${install\_path\_header}/${install\_path}

cp -r bin ${install\_path\_header}/${install\_path}/bin

cp -r lib ${install\_path\_header}/${install\_path}/lib

cp -r include ${install\_path\_header}/${install\_path}/include

cd ..

fi

export CFLAGS=""

export CPPFLAGS=""

# install gperftools

export CC=${compiler\_c}

export CXX=${compiler\_cxx}

package=gperftools-2.7

install\_path=gperftools

if [ -d ${install\_path\_header}/${install\_path} ];then

echo "${package} has been installed"

else

tar -xvf ${package}.tar.gz

tar -xvf ${package}.tar

tar -xvf ${package}.gz

cd ${package}

./configure --prefix=${install\_path\_header}/${install\_path}

make -j${compile\_cores\_number}

make install

cd ..

fi

**Install gcc as common user:**