2021/08/10

How to install the integrated code: TASK

A. Fukuyama

Professor Emeritus, Kyoto University

Preparation on macos (1)

- **Install of Xcode**
  - Xcode: development environment on macos
  - Use App Store
  - Category: Development
  - Choose and install Xcode
- **Install of Command_Line_Tools**
  - Command_Line_Tools: various Unix commands for development
  - Input command on terminal
  - `xcode-select --install`
- **Install of XQartz**
  - Download and install XQartz from https://www.xquartz.org
- **Install of java**
  - Download and install java from https://www.java.com/en/download/mac_download.jsp

Preparation on macos (2)

- **Install of Macports**
  - Download latest macports binary from https://www.macports.org
  - tab: Installing MacPorts
  - Quickstart: download and install MacPorts installer for appropriate macos version
  - MacPorts is mostly installed at /opt/local
  - Update to latest Macports `sudo port selfupdate`
- **Install of compilers**
  - gfortran: `sudo port install gcc11`
  - mpich: `sudo port install mpich`
  - others: `sudo port install gmake cmake imake`

Preparation on macos (3)

- **Install of MPI and parallel matrix solver library PETSc**
  - MPICH, blas/lapack, scalapack, metis, parmetis, MUMPS are downloaded by git and installed during the configure process of PETSc
- **Make PETSc directory and change its owner**
  - `sudo mkdir /opt/petsc`
  - `sudo chown /opt/petsc $USERNAME`
  - `cd /opt/petsc`
- **Download of latest PETSc library package by git**
  - **First download of PETSc source**
    - `git clone -b release https://gitlab.com/petsc/petsc.git petsc`
  - **In order to update PETSc source**
    - `git pull`

Preparation on macos (4)

- **Provide environment variables for PETSC in ~/.zprofile**
  - `export PETSC_DIR=/opt/petsc`
  - `export PETSC_ARCH=macports`
- **Configure script in python**
  - **Copy `macports.py` to `/opt/petsc`**
  - **Provide exec attribute to `macports.py`**
    - `chmod 755 macports.py`
  - **Execute configuration script** (It may take one hour.)
    - `./macports.py`
  - **Additional libraries are created in macports/externalpackages**
- **Make and check PETSc library**
  - `make` (It may take half an hour.)
  - `make check`

Install TASK (1)

- **Check git available**: just command input “git”
- **Set your identity**: Who changed the code?
  - git config - -global user.name “[your-full-name]”
  - git config - -global user.email [your-mail-address]
  - For example,
    - git config - -global user.name “Atsushi Fukuyama”
    - git config - -global user.email fukuyama@nucleng.kyoto-u.ac.jp
  - Data is saved in $HOME/.gitconfig
- **Create a working directory**: any directory name is OK
  - mkdir git
  - cd git

Install TASK (2)

- **Download TASK and necessary libraries** for download only
  - git clone https://git@bpsi.nucleng.kyoto-u.ac.jp/pub/git/gsaf.git
  - git clone https://git@bpsi.nucleng.kyoto-u.ac.jp/pub/git/bpsd.git
  - git clone https://git@bpsi.nucleng.kyoto-u.ac.jp/pub/git/task.git
- **Download TASK and necessary libraries** for download and upload
  - git clone ssh://username@bpsi.nucleng.kyoto-u.ac.jp/pub/git/gsaf.git
  - git clone ssh://username@bpsi.nucleng.kyoto-u.ac.jp/pub/git/bpsd.git
  - git clone ssh://username@bpsi.nucleng.kyoto-u.ac.jp/pub/git/task.git
- **Three directories are created**
  - **gsaf**: graphic library
  - **bpsd**: data interface library
  - **task**: main TASK directory

How to use git (1)

- **Repositories**
  - **local**: in your machine
  - **remotes**: in remote servers
  - **remotes/origin**: in default server: bpsi.nucleng.kyoto-u.ac.jp
- **Branches**
  - **There are several branches for code development**
    - **master**: default, stable version, often rather old
    - **develop**: latest version, where I am working
    - **others**: branches for working specific modules
  - cd task
  - git branch : list branch names, local only
  - git branch -a : list branch names, local and remote

How to use git (2)

- **To use develop branch**
  - **Create local branch develop and associate it with remote develop**
  - git checkout -t -b develop origin/develop
  - git branch
- **Change working branch**
  - git checkout master
  - git checkout develop
- **Update working branch**: download from remote repository
  - git pull
    - Your modification is kept, if committed.
    - If uncommitted modification remains, no overwrite.
    - use “git stash” to keep away your modification.
    - If there is a conflict with your committed modification, conflict are indicated in the file. Corrects them and “git pull” again.

How to use git (3)

- **To check your modification**
  - git status
- **To commit your modification with message**: update local depository
  - git commit -a -m'*message*'
- **To list all modification**
  - git log
- **To show difference from committed repository**
  - git diff [*filename*]
- **For more detail, visit**
  - https://git-scm.com/documentation

Install TASK (3)

- **Install graphic library GSAF** (start from directory git)
  - cd gsaf/src
  - cp ../arch/macosxi-gfortran64/Makefile.arch .
  - **Edit Makefile.arch**: adjust BINPATH and LIBPATH to available paths
  - make
  - make install : if necessary use “sudo make install”
  - cd test
  - make
  - ./bsctest
  - 5
  - c
  - m: CR to change focus to original window
  - e
  - cd ../../..

Install TASK (4)

- **Setup make.header file**
  - cd task
  - cp make.header.org make.header
  - **Edit make.header** to remove comments for target OS and compiler
- **Compile data exchange library BPSD**
  - cd ../bpsd
  - make
  - cd ../task
- **Compile TASK**: eq for example
  - cd eq
  - make
  - ./eq

Install TASK (5)

- **Setup matrix solver library**
  - cd mtxp
  - cp make.mtxp.org make.mtxp
  - **Edit make.mtxp** to remove comments for your configuration
  - make
- **Type of configuration**
  - no MPI: only direct band matrix solver and an iterative solver
  - with MPI: only direct band matrix solver and an iterative solver
  - with MUMPS: parallel direct solver
  - with PETSc: various parallel iterative and direct solvers
- **Compile modules**:
  - Edit the beginning of Makefile: Select matrix solver
    - **Real** matrix equation (fp,ti): any mtxp library
    - **Complex** matrix equation (wm,wf2d,wf3d): band matrix or MUMPS

How to use GSAF

- **At the beginning of the program**
  - **Set graphic resolution** (0: metafile output only, no graphics)
  - **commands**
    - **c**: continue
    - **f**: set metafile name and start saving
- **At the end of one page drawing**
  - **commands**
    - **c** or **CR**: change focus to original window and continue
    - **f**: set metafile name and start saving
    - **s**: start saving and save this page
    - **y**: save this page and continue
    - **n**: continue without saving
    - **d**: dump this page as a bitmap file “gsdump*n*”
    - **b**: switch on/off bell sound
    - **q**: quit program after confirmation

Graphic Utilities

- **Utility program**
  - **gsview**: View metafile
  - **gsprint**: Print metafile on a postscript printer
  - **gstoeps**: Convert metafile to eps files of each page
  - **gstops**: Convert metafile to a postscript file of all pages
  - **gstotgif**: Convert metafile to a tgif file for graphic editor tgif
  - **gstotsvg**: Convert metafile to a svg file for web browser
- **Options**
  - **-a**: output all pages, otherwise interactive mode
  - **-s ps**: output from page ps
  - **-e pe**: output until page pe
  - **-p *np***: output contiguous *np* pages on a sheet
  - **-b**: output without title
  - **-r**: rotate page
  - **-z**: gray output

Typical File Name of TASK

- `XXcomm.f90`: Definition of global variables, allocation of arrays
- `XXmain.f90`: Main program for standalone use, read XXparm file
- `XXmenu.f90`: Command input
- `XXinit.f90`: Default values (may still include XXparm.f90)
- `XXparm.f90`: Handling of input parameters
- `XXprep.f90`: Initialization of run, initial profile
- `XXexec.f90`: Execution of run
- `XXgout.f90`: Graphic output
- `XXfout.f90`: Text file output
- `XXsave.f90`: Binary file output
- `XXload.f90`: Binary file input

Typical input command

- When input line includes `=`, interpreted as a namelist input (e.g., `rr=6.5`)
- When the first character is not an alphabet, interpreted as line input
- `r`: Initialize profiles and execute
- `c`: Continue run
- `p`: Namelist input of input parameters
- `v`: Display of input parameters
- `s`: Save results into a file
- `l`: Load results from a file
- `q`: End of the program
- **Order of input parameter setting**
  - Setting at the subroutine `XXinit` in `XXinit.f90`
  - Read a namelist file `XXparm` at the beginning of the program
  - Setting by the input line

Install on Ubuntu (1)

1. **Install of required modules**

```
sudo apt-get install gfortran-11
sudo apt-get install gcc-11
sudo apt-get install g++-11

sudo apt-get install emacs
sudo apt-get install git
sudo apt-get install xorg-dev
sudo apt-get install valgrind
sudo apt-get install cmake
sudo apt-get install python
```

2. **Install of MPICH**

   (a) **Download the latest mpich-n1.n2.n3.tar.gz from ttypehttp://www.mp**

   (b) **Expand at /opt/mpich**

   (c) **Configure by executing "./run"**

run:

```
CC=gcc-11 CFLAGS=''-m64'' CXX=g++-11 CXXFLAGS=''-m64'' F
FFLAGS=''-m64'' ./configure --prefix=/usr/local/mpich-g
--enable-cxx --enable-fast --enable-romio --disable-sha
```

3. **Compile and install**

   (a) `make`

   (b) `make install`

Install on Ubuntu (2)

1. **Install of PETSc**

   (a) Download of petsc-3.11.3.tar.gz (*See 2.1*)

   (b) Expand at /opt/PETSc/ (*See 2.1*)

   (c) Setup environment variables (*See 2.2*)

```
export PETSC_DIR=/opt/PETSc/petsc-3.11.3
export PETSC_ARCH=gfortran
```

   (d) Configure by gfortran.py (*See 2.2*)

   (e) Execute fortran.py

gortran.py:

```
#!/usr/bin/env python

# Build PETSc, with gfortran

configure_options = [
  '--with-mpi=1',
```

```
    '--with-mpi-dir=/usr/local/mpich331-gfortran-gcc8'
,
    '--with-shared-libraries=0',
    '--with-cxx-dialect=C++11',
    '--download-mpich=0',
    '--download-hypre=0',
    '--download-fblaslapack=1',
    '--download-spooles=1',
    '--download-superlu=1',
    '--download-metis=1',
    '--download-parmetis=1',
    '--download-superlu_dist=1',
    '--download-blacs=1',
    '--download-scalapack=1',
    '--download-mumps=1'
    ]

if __name__ == '__main__':
    import sys,os
```

```
sys.path.insert(0,os.path.abspath('config'))
import configure
configure.petsc_configure(configure_options)
```