

How to install the integrated code: TASK

A. Fukuyama

Professor Emeritus, Kyoto University

- 1. Preparation for macOS
- 2. In case of macOS major version up
- 3. Preparation for Ubuntu
- 4. Introduction to git
- 5. Install task and related libraries
- 6. Introduction to the task code
- 7. Install PETSc

Preparation for macOS (1)

Install Xcode

- Xcode: development environment on macOS
- Use App Store
- Category: Development
- Choose and install Xcode

Install Command_Line_Tools

- Command_Line_Tools: various Unix commands for development
- Corresponding to the version of Xcode
- xcode-select --install
- or download from Apple Developer site

Install XQartz

- Download and install XQartz from https://www.xquartz.org
- Install java (when necessary)
 - https://www.java.com/en/download/mac_download.jsp

Preparation for macOS (2)

Install Macports

- Macports site: https://www.macports.org
- Select tab: Installing MacPorts
- Download the MacPorts installer for appropriate macOS version
- Run the MacPorts installer

Set environmental variables in \$HOME/.zprofile

- Add export PATH=/opt/local/bin:\$PATH in .zprofile
- Add export MANPATH=/opt/local/man:\$MANPATH in .zprofile

Install compiler and related modules

- sudo port install gmake cmake imake gcc13 mpich
- sudo port select gcc mp-gcc13
- sudo port select mpi mpich-mp-fortran

Preparation for macOS (3)

- To update Macports: (If rsync is available)
 - sudo port selfupdate
 - sudo port upgrade outdated
- If rsync is not available, port selfupdate will hang up.
 - See https://trac.macports.org/wiki/howto/SyncingWithGit
 - Create the source list
 - Download the source list in MacPorts server by git clone
 - Modify macports config to read the source list
 - Find outdated ports and upgrade them
 - Update the source list by git pull
 - Sync installed with the source list and select outdated (takes long)
 - sudo port upgrade outdated

In case of macOS major version-up

- macOS major version-up: e.g. $14.3 \implies 15.0$
- Update Xcode and Command_Line_Tools for the new version
- Install a new Macports binary for the new version
- Migrate Macports: make ports list, uninstall ports, reinstall ports
 - port -qv installed > myports.txt
 - port echo requested | cut -d ' ' -f 1 | uniq > requested.txt
 - sudo port -f uninstall installed
 - sudo port reclaim
 - curl -L -O https://github.com/macports/macports-contrib/raw/master/restore_ports/restore_ports.tcl
 - chmod +x restore_ports.tcl
 - xattr -d com.apple.quarantine restore_ports.tcl
 - sudo ./restore_ports.tcl myports.txt
 - sudo port unsetrequested installed
 - xargs sudo port setrequested < requested.txt</pre>
- See more detail in https://trac.macports.org/wiki/Migration

Preparation for Ubuntu

1. Install required modules

```
sudo apt-get install gfortran-13
sudo apt-get install gcc-13
sudo apt-get install g++-13
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
sudo apt-get install mpich
```

How to use git (1)

git: version and remote repository control facility

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 are conflicts with your committed modification, the conflicts are indicated in the file. Correct them and git pull again.

How to use git (3)

- To check your modification
 - git status
- To commit your modification with message: only local depository is updated. message is required.
 - 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 (1)

- Check availability of git: just command input "git"
- Set your identity: To record 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://bpsi.nucleng.kyoto-u.ac.jp/pub/git/gsaf.git
 - git clone https://bpsi.nucleng.kyoto-u.ac.jp/pub/git/bpsd.git
 - git clone https://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
 - username@ can be omitted if the usernames at remote and local are same.

Three directories are created

- gsaf: graphic library
- bpsd: data interface library
- task: main TASK directory

Install TASK (3)

Install graphic library GSAF

- cd git/gsaf/src
- Copy Makefile.arch appropriate for your environment
 - for macOS: cp ../arch/macos-gfortran/Makefile.arch .
 - for Ubuntu: cp ../arch/ubuntu-gfortran64-static/Makefile.arch .
- Edit Makefile.arch: adjust BINPATH and LIBPATH to available ones
 - BINPATH: graphic commands are located, should be included in \$PATH in ~/.profile or ./zprofile
 - LIBPATH: graphic libraries are located, should be included in library path for compiling applications using the graphic libs.
- make
- make install
 - if BINPATH is protected, use "sudo make install"

Install TASK (4)

Check the availability of GSAF library

- cd test
- make
- Applications using GSAF library must be started from X11 window such as xterm, not from Terminal on macOS.
- ./bsctest
- 5 : Choose the size of window
- c : Continue the run
- m
- New graphic window opens and marks and lines are drawn.
- To go back to the original window, enter CR.
- If focus does not change, click the original window and check XQartz preferences.
- e : Close the graphic window
- cd ../../..

Install TASK (5)

- Use the latest develop branch and setup files, make.header and make.mtxp
 - cd task
 - git checkout -t -b develop origin/develop
 - 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

Install TASK (6)

Choice of matrix solver configuration

- Single processing without MPI: make.mtxp.nompi
- Single processing with MPI: make.mtxp.mpi
- Multi processing with parallel real matrix solver: make.mtxp.petsc
- Multi processing with parallel real and complex solver: make.mtxp.petsc+mumps

Module using parallel matrix solver

- Real: fp, ti, pic, t2
- Complex: wmx, wf2d, wf2dt, wf2dx, wq

Setup matrix solver library

- cd mtxp
- cp make.mtxp.XXX make.mtxp
- make
- cd ..

Install TASK (7)

• Compile modules:

- cd lib
- make
- cd ..
- Compile and run TASK module: eq for example
 - cd eq
 - make
 - ./eq
 - **-** 5
 - C
 - r
 - **–** g
 - s, CR, CR, ⋯
 - X
 - q

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 "gsdumpn"
 - 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
- xxparm.f90: Read input parameters
- xxview.f90: Show 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
- 1: Load results from a file
- q: End of the program
- Order of input parameter setting
 - Setting at the subroutine xx_init in xxinit.f90
 - Read a namelist file xxparm at the beginning of the program
 - Setting by the input line

Install PETSc (1)

- PETSc: Parallel matrix solver library
 - blas,lapack: matrix solver tolls
 - scalapack, metis, parmetis, blacs, superlu: parallel solver tools
 - MUMPS: Direct matrix solver for real and complex
 - PETSc: Iterative matrix solver for real or complex
- Make PETSc directory and change its owner
 - sudo mkdir /opt/PETSc
 - sudo chown /opt/PETSc \$USERNAME
 - cd /opt/PETSc
- Download 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

Install PETSc (2)

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