# User Guide of V-Sphere

**Ver. 1.2.2**

**Functionality Simulation and Information Team**
**VCAD System Research Program**
**RIKEN**

2-1, Hirosawa, Wako, 351-0198, Japan

http://vcad-hpsv.riken.jp/

June 2011

| | | | |
|---|---|---|---|
| First Edition | version 1.0.0 | 5 Mar. | 2009 |
| | version 1.1.0 | 5 Jan. | 2010 |
| | version 1.2.0 | 10 May | 2011 |
| | version 1.2.1 | 6 June | 2011 |
| | version 1.2.2 | 20 June | 2011 |

# 目次

# 第 1 章

# イントロダクション

物理現象の解析や工業製品の設計のため，シミュレーションは不可欠な技術となっている．特に，製品開発ではコストや性能の検討のため，高精度な計算結果を迅速に得たいというエンジニアの要求が高くなっている．また，複雑な物理現象の解明のため，異なる物理現象の連成解析や非線形マルチスケール現象を扱うシミュレーション研究が進んでいる．これらの先端的な研究は，必然的に分散並列の大規模解析となる傾向にある．従来から，流体解析や構造解析では領域分割型の並列計算法が研究され，並列化粒度の大きな効率のよい計算手法が提案されてきた．これらは MPI を用いた並列プログラムであるが，デバッグを含めたコード開発とメンテナンスが難しい点が問題であり，開発支援の仕組みが必要となっている．この問題点を緩和する目的のために，オブジェクト指向プログラミング（Object-Oriented Programming, OOP）によるクラスライブラリ，フレームワークなどの並列プログラミング支援環境が研究されてきた．

この章では, OOP を用いて構築された様々な物理現象シミュレータ開発のためのフレームワーク V-Sphere の概要について述べる．

## 1.1　V-Sphere とソルバークラス

複数の物理ソルバを連成したマルチフィジックス解析では，複数ソルバの実行制御・管理の問題が顕在化している．物理シュミレーションの数値解法は，各々の現象に対して効率のよい固有の解法として発展し，専門性の強い研究領域を形成している．一人の研究者で全ての現象解析をカバーすることは難しいため，連成解析のシステム開発にあたり，研究者間のコラボレーションを促進する仕組みには大きな期待が寄せられている．具体的には，プログラム開発のガイドライン，あるいは共通機能を持つフレームワークを利用することにより，開発効率やプログラムの品質の向上を図ることができる．特に，大規模なソフトウェア，複数のプログラマによる協同作業の場合には，作業効率やメンテナンス性の点からはこれらの仕組みが必須である．

プログラムの開発支援と実行管理の問題点に対して，著者らは物理シミュレーションのひな型の提供と複数のソルバーコードを管理し，選択・連成実行可能なアプリケーションの機能を持つオブジェクト指向フレームワーク V-Sphere を提案している [1–3]. このフレームワークは，オブジェクト指向技術の積極的な利用により，非定常/定常物理シミュレーションのソフトウェア構造の標準化・統一化を推進している．また，アプリケーションの開発効率化・高品質化・メンテナンス性の向上に貢献し，先端シミュレーション技術の迅速なパッケージングが期待できる．

V-Sphere は，非定常物理シミュレーションのフレームワークとして設計されている．V-Sphere フレームワークは，**図 1.1** に示す様々なライブラリ機能と**図 1.2** に示す非定常物理現象のシミュレータに共通する制御構造をソルバー開発者に提供する．つまり，時間的に変化する物理現象の解析プログラムはどれも同様に記述できる点に着目し，処理の大まかな流れ（前処理，本計算，後処理の３つのステージ）を規定し，共通機能を抽出し，API としてまとめている．制御構造は SklSolverBase クラスに内包されており，開発者はこのソルバー基底クラスを継承したクラスを作成し，この派生クラスにユーザ関数・サブルーチンをクラスメソッドとして実装することによりプログラムを作成する．また，通常の意味でのサブルーチンも多く用意し，計算パラメータなどの入力データ，計算結果の入出力などの機能を提供している．これらの利用により，ソルバ開発者にとって本質的でないプログラミングを減らし，開発の効率化が期待できる．システムの実装はオブジェクト指向に基づき C++ で記述されているが，従来資産の移行，物理コードの開発者が C や Fortran 言語を使い慣れていることを考慮し，プログラミングモデルとしては手続き型言語の記述を基本としている．具体的には，Fortran/C 言語へのインタフェースを備えている．

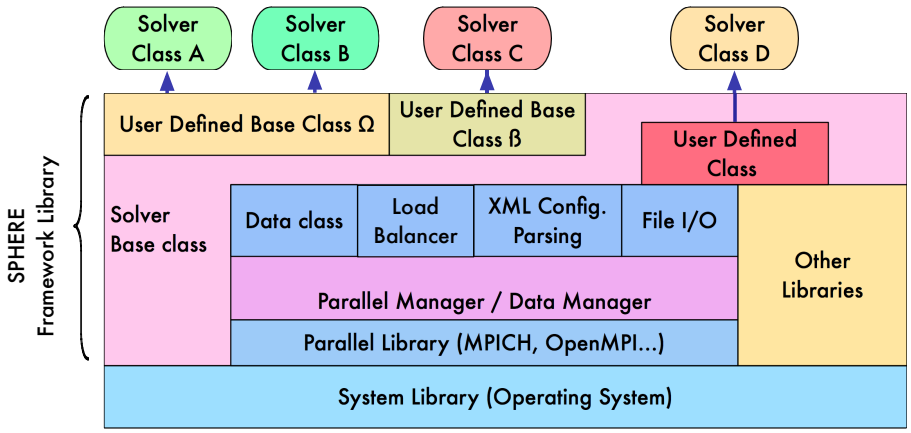


図 1.1　V-Sphere framework のブロック図．V-Sphere は様々な機能，たとえば並列ライブラリ，ファイル入出力，XML 記述によるパラメータハンドリングなどを内包する．

V-Sphere は，非定常物理シミュレーションのソルバー開発を支援する抽象度の高い汎用的なプログラム部品群をクラスライブラリの形式で提供する．それらのうち主要なクラスは SklSolverBase クラスに既に組み込まれている．例えば，ファイル入出力，ソルバー制御・物理・境界条件パラメータの読み込みと保持，ボクセルデータの前処理，境界条件の制御などの機能があり，プログラムに対するユーザーインターフェイスを規定する役割を果たす．

V-Sphere の機能を用いて作成したアプリケーションはソルバークラスとして V-Sphere 自身に登録することができ，

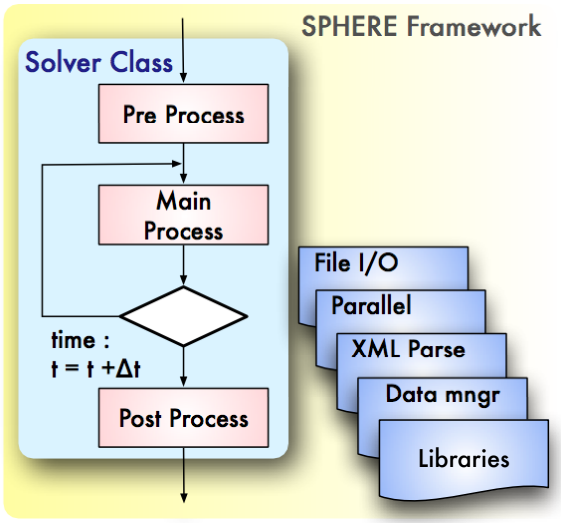


図 1.2　V-Sphere の制御構造．プリ，メイン，ポストの処理プロセスが組み込まれており，提供されるライブラリ機能を用いてソルバークラスを構築する．

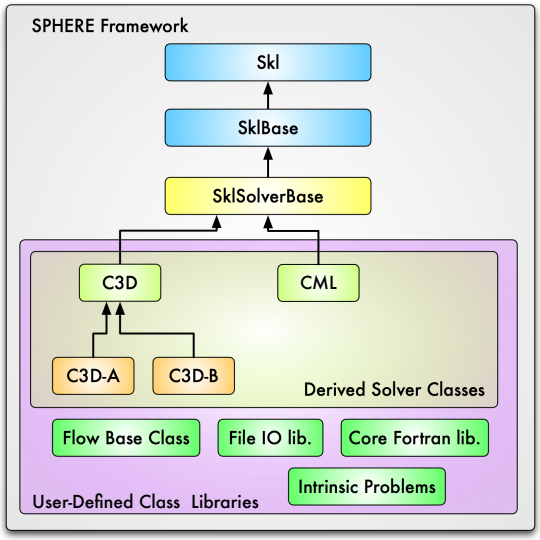


図 1.3　差分プログラミングによるソルバークラスの開発．SklSolverBase クラスから派生させて目的のソルバークラスを作成する．ソルバークラス C3D-A は C3D から派生しており，必要な機能だけが追加でプログラミングされる．また，必要に応じて，ユーザが定義したクラスライブラリに共通機能をまとめて利用できる．

登録されたソルバークラス群は共通のユーザインターフェイスを備えたアプリケーションとして振る舞う．これはエンドユーザから見ると，利用しやすいアプリケーション群と認識されるであろう．V-Sphere では，**図 1.3** に示すように Skl クラス，SklBase クラス，および SklSolverBase クラスを提供する．SklSolverBase クラスから SklSolverFB クラスと SklSolverStruct クラスが派生し，さらに SklSolverFB クラスからは各ソルバクラスが派生している．**図 1.3** には，ソルバークラス C3D から派生した 2 つのソルバークラスを示している．これらの 2 つのソルバークラス C3D-A, C3D-B は，基本的な機能は C3D クラスの機能を持つが，例えば，シミュレートする物理現象や形状近似度，変数配置などが異なるソルバーと考えることができる．異なるソルバーであっても，同じ基底クラスを利用してアプリケーションを開発することによって，ユーザーインターフェイスが統一されたアプリケーションとして構築することができる．

具体的なアプリケーションを構築する場合には，V-Sphere が提供する基本的な機能部品を用いて，より具体的な機能を構成する必要がある．つまり，ソルバークラスの詳細な処理の記述はユーザに任されている．そこで，適用範囲を限定しながらもある程度の汎用性をもつクラスを作成する必要がある．これには，二種類の方法がある．**図 1.1** において，ユーザー定義クラスやユーザー定義基底クラスがそれに相当する．ユーザー定義クラスは，文字通り，ユーザーが作成したクラスである．ユーザー定義クラスは通常のオブジェクト指向プログラミングに従い，SklSolverBase クラスの中でインスタンスして利用することができる．一方，ユーザー定義基底クラスは，SklSolverBase クラスを派生させたクラスで，これからさらに派生させたソルバークラスでの利用を想定している．つまり，ソルバー開発者はユーザー定義基底クラスを派生させて具体的なソルバークラスを作成する．V-Sphere を用いたコード開発では，両方のアプローチを利用している．

シミュレーションプログラムの利用価値を高めるためには，コードのポータビリティを始めとして，開発の効率化支援と共にメンテナンス性や利用環境なども考慮する必要がある．この観点から，V-Sphere は開発者とエンドユーザーの双方に利便性を提供する．

- 開発の効率化

  フレームワークの利用により，上位概念でプログラミングができる．つまり，アプリケーション開発者はアルゴ

リズムの記述に専念でき，またプログラムのメンテナンスが簡単になる．他方，デメリットとしてはソルバー開発者に，プログラム・データ構造やコーディングの作法を強制する面がある．

- エンドユーザーに対するインターフェイスの統一化
  利用するソルバーが異なってもアプリケーションの振る舞いは同様であるので，アプリケーション導入時のハードルは低くなる．

大規模な解析を実施する観点からは，並列化が必須となってくる．V-Sphere は，並列化も含めシミュレーションプログラム開発の効率的な開発環境を提供すること，マルチフィジックス問題を扱うことも視野に入れ複数のコードの実行管理を行うことを視野に入れている．逐次コードから並列コードへの拡張については，簡単なライブラリコールにより領域分割型の並列化に対応できるように工夫され，高い並列化性能がでることを確認している．

## 1.2　ソルバークラス

V-Sphere に移植されたソルバークラスを**表 1.1** に示す[*1]．FB クラスは，具体的な流体ソルバークラスを開発するために使うクラス群であり，単体では動作しない．

表 1.1　V-Sphere で稼働するソルバークラス

| Solver Class | Solver の説明 |
|---|---|
| CBC | 直交系の単層流三次元非圧縮非定ソルバークラス |
| FB | 流体解析に用いる基本パッケージクラス |

## 1.3　V-Sphere ユーザーガイドの構成

本ユーザーガイドの構成は以下のようになっている．2 章では MPI 通信ライブラリ，V-Sphere の環境設定，およびソルバークラスのインストールについて詳細に記述している．

[*1] 2011 年 6 月 20 日現在

# 第 2 章

# 並列ライブラリ

MPI 通信ライブラリのインストール概要について説明する．MPI 通信ライブラリは，mpich1 [4], mpich2 [5], OpenMPI [6] など，いずれでも良い．本章では mich1, mpich2, OpenMPI についてインストール方法を記す．

## 2.1　mpich1

1. 簡単な手順[*1]

   Intel Mac, Intel Compiler 11.1 の場合について示す．

```
$ ./configure --prefix=/usr/local/mpich -cc=icc -c++=icpc -fc=ifort -f90=ifort \
              -cflags=-O3 -c++flags=-O3 -rsh=ssh
$ make
$ sudo make install
```

2. トラブルシューティング

   - gcc でも icc でも可能．configure 時に環境変数を使う指示があるが，この configure はコマンドラインで指定すること．[*2]

```
--prefix=/usr/local/mpich
-fc=ifort
-f90=ifort
-rsh=rsh or ssh
```

   - fortran の設定ができていない場合には， /usr/local/mpich/bin 配下に mpif90 などのインクルードファイル，include 配下に mpif.h, f90choice/などができていないので，この点を確認する．configure の内容は，bin/mpireconfig.dat の先頭付近にコメントとして書かれている．両方を試す場合には，

```
/usr/local/mpich.gcc
/usr/local/mpich.intel
```

     などをつくり，/usr/local/mpich にスタティックリンクを張る．

```
$ cd /usr/local
$ sudo ln -s mpich.intel/ mpich
```

   - システムの設定で，場合によっては/usr/bin/mpirun がコールされる場合がある．これは PATH で /usr/local/mpich/bin を先に見るように設定しておく．

   - sphere のコンパイルで，libxml2 の pthread 関連のエラーが出ることがある．make.IA64_linux などで，XMLLIBS の最後に-lpthread を追加して対応する．

   - リンクエラーが出た場合，

```
$ nm *.a | less
```

     などで，該当の関数を探す．記号 T を全て探して，対象のアーカイブをリンクするように make.*を変更する．具体的には，FCLIBS=...

   - Linux で Intel Compiler 10.x/11.x の場合，

[*1] Mac の場合には，-rsh=ssh を使用する．Intel Compiler C/C++ 10.1 では，multibyte の文字コードの対応が不充分のようで、デフォルトの設定ではうまく動かない．このため，configure 時には次のオプションをつける．version 11.0, rev.056 以降では，-no-multibyte-chars は不要．
”-cflags=’-O3 -no-multibyte-chars’ -c++flags=’-O3 -no-multibyte-chars’”

[*2] Intel Compiler C/C++ ver.10 までは，32 ビット用コンパイラと 64 ビット用コンパイラが混在している．Intel64 のプラットホームで，cc(32bit), cce(64bit) の両方がある場合，先に cce を記述する．
”PATH=/usr/local/mpich/bin:/opt/intel/cce/10.1/bin:/opt/intel/cc/10.1/bin”

```
-cflags='-O3 -no-multibyte-chars' -c++flags='-O3 -no-multibyte-chars'
```

- Intel mac OSX 10.5 で，Intel Compiler 11.0/11.1 の場合，rsh は必ず ssh のこと．また，Intel Compiler 10.1 の場合には，Linux の場合と同様に”-no-multibyte-chars”が必要．

```
$ ./configure --prefix=/usr/local/mpich -cc=icc -c++=icpc -fc=ifort -f90=ifort \
              -cflags=-O3 -c++flags=-O3 -rsh=ssh
```

## 2.2　mpich2

mpich2 を例に説明する．

1. 簡単な手順

   MPICH2 の WEB サイト[*3]から，ソースをダウンロードして解凍する．

   作成されたディレクトリに入り，configure のために，次のようなスクリプトを用意し，実行する．インストールディレクトリは，/usr/local/mpich2 とする．

```
$ cat config_mpich2.sh
-----------------------------------
#!/bin/sh
export CC=icc
export CFLAGS=-O3
export CXX=icpc
export CXXFLAGS=-O3
export F77=ifort
export FFLAGS=-O3
export FC=ifort
export FCFLAGS=-O3
#
./configure --prefix=$1
-----------------------------------

$ ./config_mpich2.sh /usr/local/mpich2

$ make

$ sudo make install
```

## 2.3　OpenMPI

OpenMPI-1.3.2 を例に説明する．

1. 簡単な手順

   configure のために，次のようなスクリプトを用意し，実行する．インストールディレクトリは/usr/local/ompi とする．

```
$ cat config_ompi.sh
------------------------------
#!/bin/sh
export CC=icc
```

[*3] http://www.mcs.anl.gov/research/projects/mpich2/
2011 年 5 月 12 日現在，安定リリース版は mpich2-1.3.2p1.tar.gz である．

```
export CFLAGS=-O3
export CXX=icpc
export CXXFLAGS=-O3
export F77=ifort
export FFLAGS=-O3
export FC=ifort
export FCFLAGS=-O3
#
./configure --prefix=$1
------------------------------

$ ./config_ompi.sh /usr/local/ompi

$ make

$ sudo make install
```

2. PATH の設定

   実行時の mpiexec[*4]が正しいパスになっているかどうかを which コマンドで確認する．

```
$ which mpiexec
/usr/bin/mpiexec
```

   Mac OSX の場合には上記のように，デフォルトでインストールされている OpenMPI の方を見に行くので，インストールした OpenMPI の PATH を最初の方に書いておく．

```
$ cat .bash_priofile

#! /bin/sh
PATH=~/bin:~/bin/script:/usr/local/ompi/bin:${PATH}; export PATH
...
. ~/.bashrc
```

[*4] mpirun でも動く．

# 第 3 章

# V-Sphere

この章では，V-Sphere のインストール概要について説明する．

## 3.1　インストール

V-Sphere のインストールは，MPI 通信ライブラリのインストールが終了したあとに行う．V-Sphere は，単精度版と倍精度版を別々に用意する必要がある．

1. configure[*1]

**■コマンドライン**　コマンドラインで作業する場合には，次のようにタイプする．インストールディレクトリには/usr/local/sphere/を指定している．もし，/usr/local/領域へのアクセス権限がない場合には，各ユーザが書き込めるところを指定する．また，LDFLAGS には適切なパスを指定する．

```
$ ./configure --prefix=/usr/local/sphere \
              --with-comp=INTEL \
              --with-ompi=/opt/openmpi  \
              CC=icc \
              CFLAGS='-O3'\
              CXX=icpc \
              CXXFLAGS='-O3'\
              FC=ifort \
              FCFLAGS='-O3'\
              F90=ifort \
              F90FLAGS='-O3'\
              LDFLAGS=-L/opt/intel/Compiler/11.1/089/lib
```

configure でインストール先を指定すると，指定したインストールディレクトリはソースディレクトリの sphconfig/sph-cfg.xml に記録される．このファイルは, インストールディレクトリの config/sph-cfg.xml にコピーされ，sphPrjTool の reset コマンドで参照される．このため，一旦インストールディレクトリを指定したら，移動したりリネームすると，reset コマンドが正しく機能しなくなるので注意する．

**■倍精度モジュール**　倍精度計算をする場合には，V-Sphere は倍精度モジュールとしてコンパイルする必要がある．configure 時のオプションに--with-real=double を追加する．このオプションにより，C/C++ コンパイラに-DREAL_IS_DOUBLE，Fortran コンパイラには-r8 がコンパイルオプションとして自動的に追加される．

**■シェル**　次のインストールシェルは，引数としてインストールディレクトリを指定する．

```
$ configure.sh /usr/local/sphere
```

```
-------------------------------------
configure.sh (mpich + float)
-------------------------------------
#! /bin/sh
./configure --prefix=$1 \
            --with-comp=INTEL \
            --with-mpich=/usr/local/mpich \
            CC=icc \
            CFLAGS=-O3\
            CXX=icpc \
            CXXFLAGS=-O3\
```

[*1] ここでは，MacOSX，IA-64 モード，コンパイラの環境として，/opt/intel/Compiler/11.1/089 のディレクトリを仮定，コンパイルオプションとして，-O3 を指定．icc, icpc, ifort へのパスは既に指定していると仮定する．特別な仕様のマシン向けのインストールには，makefile.spec を使う．これは，特別な仕様のマシン向け（p4-dev）である．BlueGene/L の項や V-Sphere のマニュアルを参照のこと．アーキテクチャに相応しいコンパイルオプションがわからない場合は，「CXXFLAGS=-O3」「F90FLAGS=-O3」だけでも可．Intel Compiler C/C++ version 10 は，multibyte の文字コードの対応が不充分のため，C/C++ のコンパイルオプションで-no-multibyte-chars を明示的に指示する．Mac 用の Intel Compiler 11.0 では不要だが，Linux では必要．

```
            FC=ifort \
            FCFLAGS=-O3\
            F90=ifort \
            F90FLAGS=-O3\
            LDFLAGS=-L/opt/intel/Compiler/11.0/059/lib

-------------------------------------
configure.sh (mpich + double)
-------------------------------------
#! /bin/sh
./configure --prefix=$1 \
            --with-comp=INTEL \
            --with-real=double \
            --with-mpich=/usr/local/mpich \
            CC=icc \
            CFLAGS=-O3\
            CXX=icpc \
            CXXFLAGS=-O3\
            FC=ifort \
            FCFLAGS=-O3\
            F90=ifort \
            F90FLAGS=-O3\
            LDFLAGS=-L/opt/intel/Compiler/11.0/059/lib

-------------------------------------
configure.sh (OpenMPI + float)
-------------------------------------
#! /bin/sh
./configure --prefix=$1 \
            --with-comp=INTEL \
            --with-ompi=/usr/local/ompi \
            CC=icc \
            CFLAGS=-O3\
            CXX=icpc \
            CXXFLAGS=-O3\
            FC=ifort \
            FCFLAGS=-O3\
            F90=ifort \
            F90FLAGS=-O3\
            LDFLAGS=-L/opt/intel/Compiler/11.0/059/lib

-------------------------------------
configure.sh (OpenMPI + double)
-------------------------------------
#! /bin/sh
./configure --prefix=$1 \
            --with-comp=INTEL \
            --with-real=double \
            --with-ompi=/usr/local/ompi \
            CC=icc \
            CFLAGS=-O3\
            CXX=icpc \
            CXXFLAGS=-O3\
            FC=ifort \
            FCFLAGS=-O3\
            F90=ifort \
            F90FLAGS=-O3\
            LDFLAGS=-L/opt/intel/Compiler/11.0/059/lib
```

2. make [*2]

```
$ make
```

[*2] make 時に libimf.so が見つからないなどのメッセージが出る場合は，ユーザの LD_LIBRARY_PATH にパス を加えておく．
”LD_LIBRARY_PATH=/opt/intel/Compiler/11.0/056/lib:/usr/local/mpich/lib”

3. Install

```
$ sudo make install   または   make install
```

V-Sphere ライブラリがインストールディレクトリにインストールされると，次に示すようなコンパイルに関する情報がソースディレクトリの sphconfig/sph-cfg.xml に記述される．

```
<SphereEnvironment>
  <Param name="SPHEREDIR" dtype="STRING" value= "/usr/local/sphere"/>
  <Param name="CXX" dtype="STRING" value="icpc"/>
  <Param name="CXXFLAGS" dtype="STRING" value="-O3"/>
  <Param name="CC" dtype="STRING" value="icc"/>
  <Param name="CFLAGS" dtype="STRING" value="-O3"/>
  <Param name="FC" dtype="STRING" value="ifort"/>
  <Param name="FCFLAGS" dtype="STRING" value="-O3"/>
  <Param name="F90" dtype="STRING" value="ifort"/>
  <Param name="F90FLAGS" dtype="STRING" value="-O3"/>
  <Param name="LDFLAGS" dtype="STRING" value="-L/opt/intel/Compiler/11.1/067/lib"/>
  <Param name="SPH_DEVICE" dtype="STRING" value="Snow_Leopard"/>
  <Param name="MPICH_DIR" dtype="STRING" value="/usr/local/ompi"/>
  <Param name="MPICH_CFLAGS" dtype="STRING" value="-I/usr/local/ompi/include"/>
  <Param name="MPICH_LDFLAGS" dtype="STRING" value="-L/usr/local/ompi/lib"/>
  <Param name="MPICH_LIBS" dtype="STRING" value="-lmpi"/>
  <Param name="XML2FLAGS" dtype="STRING" value="-I/usr/include/libxml2"/>
  <Param name="XML2LIBS" dtype="STRING" value="-lxml2 -lz -lpthread -licucore -lm"/>
  <Param name="SPHERE_CFLAGS" dtype="STRING" value="-DSKL_TIME_MEASURED -D_CATCH_BAD_ALLOC
               -I/usr/local/vsph175/include"/>
  <Param name="SPHERE_LDFLAGS" dtype="STRING" value="-L/usr/local/vsph175/lib"/>
  <Param name="SPHERE_LIBS" dtype="STRING" value="-lsphapp -lsphbase -lsphls -lsphfio -lsphdc
               -lsphcrd -lsphcfg -lsphftt -lsphvcar"/>
  <Param name="LIBS" dtype="STRING" value="-lifport -lifcore"/>
</SphereEnvironment>
```

**■マニュアルインストール**　別の方法として，Config.spec を編集する．ポイントは，libxml2 と install コマンドのパス．

```
$ xml2-config --libs
```

を実行してメッセージが返れば OK.

```
$ which install
```

インストールコマンドがあれば OK. その後,

```
$ make -f Makefile.spec
# make -f Makefile.spec install
```

4. インストールに失敗する場合

make に失敗して，何度もインストールしていると Make で使用する環境変数がおかしくなることがある．やり直すときは、コンフィギュレーションをクリアし，最初からインストール作業を行う．

```
$ make distclean   （コンフィギュレーションのクリア）
```

ただし，再インストールの場合は tar ボールから解凍して再試行する方がより安全．また，上記の

`make distclean` を実行すると，設定ファイルが全て消去される．

## 3.2　アンインストール

V-Sphere をアンインストールする場合には，インストールしたディレクトリ（configure でオプション指定したディレクトリ）の sphere を削除する．

# 第 4 章

# プロジェクトツールを用いた開発環境の構築

この章では，ソルバークラスの開発を行うために，プロジェクトツールを用いた環境構築について説明する．ソルバーの開発を行わない，エンドユーザは本章は読み飛ばして構わない．

## 4.1　sphPrjTool を用いた開発環境の構築

本節では，V-Sphere を用いた開発を行う場合の環境を設定する．ツールとして，sphPrjTool を用いる．sphPrjTool の詳細については，V-Sphere のマニュアル 04_UtilityTools を参照のこと．また，PRJ_CBC が提供されている場合は，「CBC_UG.pdf」の「2.3.2 sphPrjTool を用いた簡単なインストール」を参照のこと．

### 4.1.1　sphPrjTool

例として，CBC ソルバークラスについて説明する．以下のようなソースツリーを想定する．

```
CBC-x.x.x
  |
  +- src                          ソースコード
      +- project_local_settings   プロジェクトのコンパイル環境設定
      +  F_CBC                    CBC クラスの Fortran ファイル
      +- F_CPC                    CPC クラスの Fortran ファイル
      +- F_VOF                    VOF クラスの Fortran ファイル
      +- FB                       FlowBase クラス（ユーザー定義クラス群）
      +- IP                       組み込み例題クラス群
```

**プロジェクトの作成とソースファイルの登録**

まず，src 直下のディレクトリで sphPrjTool を起動し，プロジェクト名とソルバーキーワードを登録，登録内容を確認してセーブする．

```
$ cd src
$ sphPrjTool
sphPrjTool> help
sphPrjTool> new -p PRJ_CBC
sphPrjTool> new -s CBC
sphPrjTool> print
sphPrjTool> save
sphPrjTool> quit
```

以下に，print の出力結果を示す．

```
sphPrjTool> print
Project Name : PRJ_CBC
  Compile Environment
    CC                  : icc
    CFLAGS              : -O3
    CXX                 : icpc
    CXXFLAGS            : -O3
    FC                  : ifort
    FCFLAGS             : -O3
    F90                 : ifort
    F90FLAGS            : -O3
    LDFLAGS             : -L/opt/intel/composerxe/lib
    LIBS                : -lifport -lifcore
    SPH_USR_DEF_LIBS    :
    UDEF_OPT            : -DTD_USE_NAMESPACE -DNON_POLYLIB -DNON_CUTLIB
    UDEF_INC_PATH       : -I../../Cutlib_2_0_0/include -I../../Polylib_2_0_2/include
    UDEF_LIB_PATH       : -L../../Polylib_2_0_2/lib -L../../Cutlib_2_0_0/lib
    UDEF_LIB_UPPER      :
    UDEF_LIB_LOWER      :
    Use Module.
        Generate parallel module.
```

```
    ---
    Reference only (Unmodifiable):
    SPHEREDIR           : /usr/local/sphere
    SPH_DEVICE          : Snow_Leopard
    MPICH_DIR           : /opt/openmpi
    MPICH_CFLAGS        : -I/opt/openmpi/include
    MPICH_LDFLAGS       : -L/opt/openmpi/lib
    MPICH_LIBS          : -lmpi
    XML2FLAGS           : -I/opt/local/include/libxml2
    XML2LIBS            : -L/opt/local/lib -lxml2 -lz -lpthread -liconv -lm
    SPHERE_CFLAGS       : -DSKL_TIME_MEASURED -D_CATCH_BAD_ALLOC -I/usr/local/vsph184_float_64/include
    SPHERE_LDFLAGS      : -L/usr/local/vsph184_float_64/lib
    SPHERE_LIBS         : -lsphapp -lsphbase -lsphls -lsphfio -lsphdc -lsphcrd -lsphcfg
                          -lsphftt -lsphvcar
    SKL_REAL is float. (REALOPT = float)
    ---

  Regist Solver
    Name : CBC
      Regist file :
        FortranFuncCBC.h
        SklSolverCBCDefine.h
        SklSolverCBCInitialize.C
        SklSolverCBCLoop.C
        SklSolverCBCPost.C
        SklSolverCBCUsage.C
        SklSolverCBC.C
        SklSolverCBC.h
```

この作業で，作業ディレクトリでは以下のようなファイル構成になる．

```
PRJ_CBC/
   |
   +app/Makefile
   |    SklCreateSolver.C
   |    SklDeleteSolver.C
   |    SklFactoryCBC.C
   |    SklFactoryCBC.h
   |    SklSolverType.h
   |
   +bin/
   |
   +CBC/CBC.xml
   |    FortranFuncCBC.h
   |    Makefile
   |    SklSolverCBC.C
   |    SklSolverCBC.h
   |    SklSolverCBCDefine.h
   |    SklSolverCBCInitialize.C
   |    SklSolverCBCLoop.C
   |    SklSolverCBCPost.C
   |    SklSolverCBCUsage.C
   |
   Makefile
   PRJ_CBC.xml               プロジェクト設定ファイル
   project_local_settings    コンパイル時に利用する環境設定ファイル
```

次に，プロジェクト設定ファイルを指定して起動する．

```
$ cd PRJ_CBC
$ sphPrjTool PRJ_CBC.xml
```

または，sphPrjTool 起動後に

```
> load PRJ_CBC.xml
```

とすると，ファイルの登録や環境設定が行える．詳細は，マニュアルおよびヘルプコマンドを参照のこと．新規にソースファイルを SOLVER_NAME にリンクする場合には，下記の操作を行う．ファイル名は必ず絶対パスか，sphPrjTool を起動したディレクトリからの相対パスを指定する．内部的には，プロジェクトディレクトリ[*1]を基点として相対パスで記述される．[*2]

```
sphPrjTool> regist -f SOLVER_NAME *.[Ch]
sphPrjTool> regist -f SOLVER_NAME *.f90
sphPrjTool> save
```

sphPrjTool を使って，プロジェクト設定ファイルを修正・保存すると，環境設定ファイル project_local_settings ファイルの内容が修正内容に応じて変更される．したがって，project_local_settings ファイルを直接編集して環境設定を行った場合は，プロジェクト設定ファイルを修正・保存すると project_local_settings ファイルの内容がリセットされてしまうので，注意すること．

### PLS ファイルの設定

project_local_settings ファイルは，上記の作業で PRJ_CBC の直下に生成される．もし，複数のプロジェクトで共通の設定を利用する場合には，src 直下に移動することも考えられる．その場合には，プロジェクトの設定で，次のように project_local_settings ファイルの読み込み先を変更する．

```
$ pwd
CBC-x.x.x/src/PRJ_CBC

$ mv project_local_settings ..

$ sphPrjTool PRJ_CBC.xml
sphPrjTool> setpls ../project_local_settings
sphPrjTool> save
sphPrjTool> quit
```

これにより，プロジェクト情報は次のように表示される．

```
sphPrjTool> print
    ...
    project_local_settings="../project_local_settings". (Relative path from Prj-Dir)
    ...
```

### 並列モジュールの指定

並列版のソルバー実行モジュールを作成する場合には，module コマンドを用いて指定する．

```
sphPrjTool> module parallel
sphPrjTool> print
    ...
    Generate parallel module.
    ...
```

[*1] プロジェクト情報が格納されるディレクトリ．つまり，プロジェクトのコンフィギュレーションファイル（プロジェクト名.xml）の存在するディレクトリ．

[*2] .[Ch] は接尾辞 C または h をもつファイルをプロジェクトにリンクし，.f90 は接尾辞 f90 をもつファイルをプロジェクトにリンクすることを意味する．

**ユーザ定義のオプション指定**

reset localsetting コマンドでリセットされないユーザが指定オプションは環境変数で指定する．詳細は V-Sphere マニュアルを参照．

```
sphPrjTool> env UDEF_*
sphPrjTool> print
   ...
   UDEF_OPT=-DTD_USE_NAMESPACE -DNON_POLYLIB -DNON_CUTLIB
   UDEF_INC_PATH=-I../../Cutlib_2_0_0/include -I../../Polylib_2_0_2/include
   UDEF_LIB_PATH=-L../../Polylib_2_0_2/lib -L../../Cutlib_2_0_0/lib
   UDEF_LIB_UPPER=
   UDEF_LIB_LOWER=
   ...
```

## 4.2　ユーザ定義の非ソルバーモジュールの導入

非ソルバーモジュールとして，ユーザ定義クラスを作成する．ユーザ定義クラスは，ソルバークラス内で使用するユーザが作成したクラス群である．

PRJ_CBC ディレクトリで sphPrjTool を起動し，非ソルバーモジュールをプロジェクトに組み込む．複数のモジュールを利用する場合には，ライブラリのリンク順を考慮する必要があるため[*3]，モジュール間の依存関係による読み込む．モジュールを登録するとき，基底クラスを後で登録する点に注意する．ここでは，CBC ≫ IP ≫ FB のような依存関係がある（左のモジュールは右に依存している）．

```
$ sphPrjTool PRJ_CBC.xml
sphPrjTool> new -o IP
sphPrjTool> new -o FB
sphPrjTool> save
sphPrjTool> quit
```

プロジェクト情報は次のように表示される．

```
sphPrjTool> print
  ...
  Regist NonSolver
    Name : IP
    Name : FB
```

次に，ソルバークラスと非ソルバーモジュールにソースファイルを登録する．

```
$ sphPrjTool PRJ_CBC.xml
sphPrjTool> regist -f IP ../IP/*.[Ch]
sphPrjTool> regist -f FB ../FB/*.[Ch]
```

または，FB.xml ファイルなどを直接編集した後，sphPrjTool で再度 PRJ_CBC.xml ファイルをロードとセーブすると変更が反映される．

```
CBC-x.x.x
  |
  +- src                          ソースコード
      +- project_local_settings   プロジェクトのコンパイル環境設定
      +  F_CBC                    CBC クラスの Fortran ファイル
```

[*3] Makefile 中の SPH_SOLV_FLAGS などの順序が重要となる．

```
+- F_CPC                    CPC クラスの Fortran ファイル
+- F_VOF                    VOF クラスの Fortran ファイル
+- FB                       FlowBase クラス（ユーザー定義クラス群）
+- IP                       組み込み例題クラス群
|
+- PRJ_CBC                          CBC プロジェクト
   +- app                  アプリケーションコンパイルディレクトリ
   +- bin                  バイナリモジュール格納ディレクトリ
   |
   +- CBC                  CBC ソルバークラスのソースファイル
   |   +- CBC.xml          CBC クラスのコンパイル環境設定
   |
   +- FB                   非ソルバークラスディレクトリ　FlowBase
   |   +- FB.xml           FB クラスのコンパイル環境設定
   |
   +- IP                   非ソルバークラスディレクトリ　組み込み例題クラス群
   |   +- IP.xml           IP クラスのコンパイル環境設定
   |
   +- Makefile             アプリケーションコンパイル用 Linux/Mac
      PRJ_CBC.xml          CBC のコンパイル設定
```

# 第 5 章

# 各種プラットホーム対応

この章では，各種プラットホームにおけるインストールについて説明する．

## 5.1 RICC

本節では，理化学研究所の RICC システム[*1]環境でのコンパイルについて示す．

RICC システムでは幾種類かのコンパイラと MPI ライブラリが利用可能なので，各ライブラリに合わせてコンパイルを行う．ここでは，OpenMPI と富士通 MPI について示す．

- OpenMPI

  V-Sphere のコンパイル前に，以下の修正を行う．

  1. libmpi_cxx.so.0 のパスの設定
     以下のコマンドを実行して，libmpi_cxx.so.0 のパスの指定を行う[*2]．

     ```
     $ export LD_LIBRARY_PATH=/usr/local/openmpi/intel/lib:$LD_LIBRARY_PATH
     ```

  2. CLTK_TARGET_MACHINE の指定
     home ディレクトリで.cltkrc というファイルを作成し，以下の情報を記述する．

     ```
     CLTK_TARGET_MACHINE = pc
     ```

  3. コンパイル
     後述のインストールシェルを用いてインストールする．シェルの引数にはインストール先のディレクトリをフルパスで指定する．RICC ではユーザは，自分の管理ディレクトリにインストールすること．

     ```
     $ configure_ic_ompi.sh INSTALL_DIR
     $ make
     $ make install
     ```

     富士通コンパイラを利用する場合には，include/SklTiming.h ファイルに，以下のインクルード文を追加する[*3]．

     ```
     #include <stdio.h>
     ```

- Fujitsu MPI

  1. CLTK_TARGET_MACHINE の指定
     home ディレクトリで.cltkrc というファイルを作成し，以下の情報を記述する．

     ```
     CLTK_TARGET_MACHINE = pc
     ```

  2. _USE_SKL_FSEEK の定義
     include/endianUtil.h ファイルに以下の記述を追加する[*4]．

     ```
     #define _USE_SKL_FSEEK
     ```

  3. -lmpi の指定の削除
     src/utility/sphDataGather/Makefile 内に記述されている-lmpi の指定を以下のようにコメントアウトする[*5]．

[*1] http://accc.riken.go.jp/ricc.html
[*2] 後述のインストールシェルを利用する場合には，シェルに書き込んであるが，/.bashrc に書いておくと良い．sphPrjTool の使用時にも必要．
[*3] V-Sphere で対応予定．
[*4] 富士通 MPI を使用時に fseek の SEEK_SET や SEEK_CUR が使用できないため．今後対応予定．
[*5] 富士通 MPI を使用時に sphDataGather の Makefile 内で lmpi についてのエラーが出るため．今後対応予定．

```
MPICH_LIBS = #-lmpi
dataGather_LDADD = -lsphcfg #-lmpi
sphDataGather_LDADD = -lsphcfg #-lmpi
```

4. コンパイル

```
$ ./configure  または　インストールシェル
$ make
$ make install
```

- インストールシェル

  コンパイラと MPI ライブラリーの組み合わせにより，次のインストールシェルが利用できる．これらのシェルの引数をディレクトリ名として V-Sphere をインストールする．どの場合にも，倍精度計算の場合には--with-real=double を追加する．

```
-------------------------------------
Intel コンパイラ +OpenMPI  configure_ic_ompi.sh
-------------------------------------
#! /bin/sh
export LD_LIBRARY_PATH=/usr/local/openmpi/intel/lib:$LD_LIBRARY_PATH
 ./configure --prefix=$1 \
             --with-comp=INTEL \
             --with-ompi=/opt/FJSVcltk/bin \
             FC="mpif77 -intel -openmpi" \
             FCFLAGS=-O3 \
             F90="mpif90 -intel -openmpi" \
             F90FLAGS=-O3 \
             CC= "mpicc -intel -openmpi" \
             CFLAGS= -middle \
             CXX= "mpic++ -intel -openmpi" \
             CXXFLAGS= -O3 \
             LDFLAGS= "-L/opt/intel/Compiler/11.1/046/lib/intel64" \

-------------------------------------
富士通コンパイラ + 富士通 MPI  configure_fc_fmpi.sh
-------------------------------------
#! /bin/sh
export LD_LIBRARY_PATH=/usr/local/openmpi/intel/lib:$LD_LIBRARY_PATH
 ./configure --prefix=$1 \
             --with-comp=FJ \
             --with-ompi=/opt/FJSVcltk/bin \
             FC="mpif77 -fj -fjmpi" \
             FCFLAGS=-O3 \
             F90="mpif90 -fj -fjmpi" \
             F90FLAGS=-O3 \
             CC= "mpicc -fj -fjmpi" \
             CFLAGS= -O3 \
             CXX= "mpic++ -fj -fjmpi" \
             CXXFLAGS= -O3 \
             LDFLAGS= "-L/opt/intel/Compiler/11.1/046/lib/intel64" \

-------------------------------------
Intel コンパイラ + 富士通 MPI  configure_ic_fmpi.sh
-------------------------------------
#! /bin/sh
export LD_LIBRARY_PATH=/usr/local/openmpi/intel/lib:$LD_LIBRARY_PATH
 ./configure --prefix=$1 \
             --with-comp=INTEL \
             --with-ompi=/opt/FJSVcltk/bin \
             FC="mpif77 -intel -fjmpi" \
             FCFLAGS=-O3 \
             F90="mpif90 -intel -fjmpi" \
             F90FLAGS=-O3 \
             CC= "mpicc -intel -fjmpi" \
```

```
            CFLAGS= -middle \
            CXX= "mpic++ -intel -fjmpi" \
            CXXFLAGS= -O3 \
            LDFLAGS= "-L/opt/intel/Compiler/11.1/046/lib/intel64" \
```

## 5.2 IBM BlueGene/L

本節では，IBM BlueGene/L でのコンパイルについて説明する．コンパイル環境は，IBM XLFortran, XLC++ コンパイラ，クロスコンパイルである．

### 5.2.1 libxml2(2.6.30)

1. configure –prefix だけ変更すること．（libxml2 インストールディレクトリ）

```
$ user@quadro:~/XML2/libxml2-2.6.30> ./configure --prefix=/gfs1/user/XML2 CC=blrts_xlc
CXX=blrts_xlC F77=blrts_xlf CFLAGS="-DLIBXML2_STATIC -O3 -qarch=440d -qtune=440
-I/bgl/BlueLight/ppcfloor/bglsys/include" CXXFLAGS="-O3 -qarch=440d -qtune=440
-I/bgl/BlueLight/ppcfloor/bglsys/include" FFLAGS="-O3 -qarch=440d -qtune=440
-I/bgl/BlueLight/ppcfloor/bglsys/include" LDFLAGS="-L/bgl/BlueLight/ppcfloor/bglsys/lib
-lmpich.rts -lmsglayer.rts -lrts.rts -ldevices.rts" --enable-shared=no --without-threads
--without-python --without-ftp --without-http --without-readline --disable-ipv6
```

2. make testapi，runtest（サンプルソース？）でリンクエラーが出る．Makefile のリンクをしている行をコメントにして make する．

```
L.702
# $(LINK) $(runtest_LDFLAGS) $(runtest_OBJECTS) $(runtest_LDADD) $(LIBS)

L.741
# $(LINK) $(testapi_LDFLAGS) $(testapi_OBJECTS) $(testapi_LDADD) $(LIBS)
```

3. make install

```
$ make install
```

### 5.2.2 V-sphere

Makefile.spec を使用して make を実行する．以下の例では，V-Sphere version 1.4.1 を用いた記述になっている．

1. Config.spec の編集
   変更箇所は以下のとおりである．[*6]

```
INSTALLDIR      = /gfs1/user/sphere/Vsphere_1_4_1_lib
CXXCOMPTYPE     = IBM
CXXDIR          = /usr
CXXCOMP         = blrts_xlC
CXXOPT          = -qarch=440 -qtune=440 -O3 -D_NON_P4_DEVICE_

CCOMPTYPE       = IBM
```

[*6] INSTALLDIR, XML2FLAGS, XML2LIBS は，ユーザ毎の設定になる．INSTALLDIR は，sphere ライブラリをインストールするディレクトリを指定すること．XML2FLAGS, XML2LIBS は，「/gfs1/user/XML2」を指定した libxml2 インストールディレクトリ（–prefix）に置き換えること．

```
CCDIR           = /usr
CCOMP           = blrts_xlc
COPT            = $(CXXOPT)

FCOMPTYPE       = IBM
FCDIR           = /usr
FCOMP           = blrts_xlf
FCOPT           = $(CXXOPT)

F90COMPTYPE     = IBM
F90DIR          = /usr
F90COMP         = blrts_xlf90
F90OPT          = $(CXXOPT)

MPICH_DIR       = /bgl/BlueLight/ppcfloor/bglsys
XML2FLAGS       = -I/gfs1/user/XML2/include/libxml2
XML2LIBS        = -L/gfs1/user/XML2/lib -lxml2
RANLIB          = echo
```

2. Makefile.spec の編集変更箇所は以下のとおり.

```
MPI_LIBS = -lmpich
         ↓
MPI_LIBS = -lmpich.rts -lmsglayer.rts -lrts.rts -ldevices.rts
```

3. make、install の実行

```
$ make -f Makefile.spec
$ make -f Makefile.spec install
```

## 5.3　AMD Opteron

本節では，AMD Opteron 上での sphere コンパイルについて述べる．コンパイラを PGI コンパイラとしている．

1. configure

```
$ ./configure --prefix=/gfs1/user/sphere/Vsphere_1_4_1_lib --with-mpich=/usr/local/mpich
FC=pgf77 FCFLAGS=-O3 F90=pgf90 F90FLAGS=-O3 CXX=pgCC CXXFLAGS="-O3 -D_NON_P4_DEVICE_"
```

2. Vcar ソース修正 SklVcarManifest.C で_ATOL のエラーが出るので，ソースの先頭付近に以下の行を追加する．
   `#define _ATOL atol`
3. make

```
$ make
$ make install
```

## 5.4　QUEST

本節では，理化学研究所の Quest システム[*7]（PFU 製 RG1000×64 台, 1024node, 1CPU/node, 2cores/CPU, 2GB/node, GE）環境でのコンパイルについて示す．Quest システムでは幾種類かの MPI ライブラリが利用可能なので，各ライブラリに合わせてコンパイルを行う．下記に，インストールシェルのサンプルを示す．

[*7] Quest システムの詳細については，VPN 経由で，http://quest.q.riken.jp

倍精度のモジュールを作成する場合には，--with-real=double を追加すること．

```
-----------------------------------
configure_mpich.sh \\ Intel Compiler + mpich
-----------------------------------
#! /bin/sh
#
# at .bashrc
#
# QUEST_COMPILER_TYPE=intel
# QUEST_MPI_TYPE=mpich
# [ -f /opt/FJSVcltk/etc/questrc.sh ] && source /opt/FJSVcltk/etc/questrc.sh
#

 ./configure --prefix=$1 \
             --with-comp=INTEL \
             --with-mpich=/usr/local/mpich/intel \
             --enable-nop4dev \
             FC=/opt/intel/Compiler/11.0/081/bin/intel64/ifort \
             FCFLAGS=-O3 \
             F90=/opt/intel/Compiler/11.0/081/bin/intel64/ifort \
             F90FLAGS=-O3 \
             CC=/opt/intel/Compiler/11.0/081/bin/intel64/icc \
             CFLAGS=-O3 \
             CXX=/opt/intel/Compiler/11.0/081/bin/intel64/icpc \
             CXXFLAGS=-O3 \
             LDFLAGS=-L/opt/intel/Compiler/11.0/081/lib/intel64 \

-----------------------------------
configure_ompi.sh \\ Intel Compiler + openmpi
-----------------------------------
#! /bin/sh
#
# at .bashrc
#
# QUEST_COMPILER_TYPE=intel
# QUEST_MPI_TYPE=openmpi
# [ -f /opt/FJSVcltk/etc/questrc.sh ] && source /opt/FJSVcltk/etc/questrc.sh
#
 ./configure --prefix=$1 \
             --with-comp=INTEL \
             --with-ompi=/usr/local/openmpi/intel \
             --enable-nop4dev \
             FC=/opt/intel/Compiler/11.0/081/bin/intel64/ifort \
             FCFLAGS=-O3 \
             F90=/opt/intel/Compiler/11.0/081/bin/intel64/ifort \
             F90FLAGS=-O3 \
             CC=/opt/intel/Compiler/11.0/081/bin/intel64/icc \
             CFLAGS=-O3 \
             CXX=/opt/intel/Compiler/11.0/081/bin/intel64/icpc \
             CXXFLAGS=-O3 \
             LDFLAGS=-L/opt/intel/Compiler/11.0/081/lib/intel64 \

-----------------------------------
configure_ompi_dbl.sh \\ Intel Compiler + openmpi, double precision
-----------------------------------
#! /bin/sh
#
# at .bashrc
#
# QUEST_COMPILER_TYPE=intel
# QUEST_MPI_TYPE=openmpi
# [ -f /opt/FJSVcltk/etc/questrc.sh ] && source /opt/FJSVcltk/etc/questrc.sh
#
 ./configure --prefix=$1 \
             --with-comp=INTEL \
             --with-ompi=/usr/local/openmpi/intel \
             --enable-nop4dev \
             --with-real=double \
             FC=/opt/intel/Compiler/11.0/081/bin/intel64/ifort \
             FCFLAGS=-O3 \
```

```
             F90=/opt/intel/Compiler/11.0/081/bin/intel64/ifort \
             F90FLAGS=-O3 \
             CC=/opt/intel/Compiler/11.0/081/bin/intel64/icc \
             CFLAGS=-O3 \
             CXX=/opt/intel/Compiler/11.0/081/bin/intel64/icpc \
             CXXFLAGS=-O3 \
             LDFLAGS=-L/opt/intel/Compiler/11.0/081/lib/intel64 \

-------------------------------------
configure_impi.sh \\ Intel Compiler + Intelmpi
-------------------------------------
#! /bin/sh
#
# at .bashrc
#
# QUEST_COMPILER_TYPE=intel
# QUEST_MPI_TYPE=intelmpi
# [ -f /opt/FJSVcltk/etc/questrc.sh ] && source /opt/FJSVcltk/etc/questrc.sh
#
 ./configure --prefix=$1 \
             --with-comp=INTEL \
             --with-ompi=/opt/intel/impi/3.2 \
             --enable-nop4dev \
             FC=/opt/intel/Compiler/11.0/081/bin/intel64/ifort \
             FCFLAGS=-O3 \
             F90=/opt/intel/Compiler/11.0/081/bin/intel64/ifort \
             F90FLAGS=-O3 \
             CC=/opt/intel/Compiler/11.0/081/bin/intel64/icc \
             CFLAGS=-O3 \
             CXX=/opt/intel/Compiler/11.0/081/bin/intel64/icpc \
             CXXFLAGS=-O3 \
             LDFLAGS=-L/opt/intel/Compiler/11.0/081/lib/intel64 \
```

## 5.5　Windows

以下の環境で V-Sphere.exe モジュールの作成を実施した．

- WindowsXP sp3 32bit
- Microsoft Visual Studio 2008
- Intel Compiler 10.1.021
- MPICH2 ver 1.0.7
- libxml2, zlib, iconv のインストールパス
  C:¥Program Files¥ext_libs¥libxml2
  C:¥Program Files¥ext_libs¥zlib
  C:¥Program Files¥ext_libs¥iconv

### 5.5.1　V-Sphere のインストール

V-Sphere の Windows インストーラの “setup_sphere.msi” を起動して V-Sphere をインストールする．デフォルト設定では，以下のフォルダ，ファイルがインストールされる．

```
C:\Program Files\sphere
 ├── bin
 │       dataGather.exe
 │       sphCfgTool.exe
 │       sphCfgTool.exe.config
 │       sphDataGather.exe
```

```
|       sphMbxTool.exe
|       sphPrjTool.exe
|
+-- config
|       sph-cfg.xml
|
+-- doc
|
+-- html
|
+-- include
|   |   DebugLog.h
|   |   endianUtil.h
|   |   Skl.h
|   |   SklBase.h
|   |   SklDefine.h
|   |   SklGloval.h
|   |   sklparaf.h
|   |   SklReservWord.h
|   |   SklSolverBase.h
|   |   SklSolvFactoryBase.h
|   |   SklTiming.h
|   |   SklUtil.h
|   |   SklVersion.h
|   |   SklXMLType.h
|   |   sph_win32_util.h
|   |   utilPath.h
|   |   vfvPathUtil.h
|   |
|   +-- base
|   +-- config
|   +-- coord
|   +-- dataclass
|   +-- fileio
|   +-- ftt
|   +-- linearsolver
|   +-- parallel
|   \-- vcar
|
\-- lib
        libsphapp.lib
        libsphbase.lib
        libsphcfg.lib
        libsphcrd.lib
        libsphdc.lib
        libsphfio.lib
        libsphftt.lib
        libsphls.lib
        libsphparadmy.lib
        libsphparampi.lib
        libsphvcar.lib
```

### 5.5.2　環境設定

プロジェクトツールによるソルバの project_local_settings，Makefile.win を作成する為に，“sphCfgTool.exe” を起動する**図 5.1**[*8]．

“sphCfgTool.exe” は、インストール中にも起動される．インストール後は，直接 “sphCfgTool.exe” を起動するか、プログラムメニュー「Sphere」-「sphCfgTool」から起動．sphCfgTool の画面で，パスの設定が異なっていると赤字**図 5.2** で “Invalid Folder” が表示される．すべてパス設定が行われていることを確認する．

[*8] C:¥Program Files¥sphere¥bin¥sphCfgTool.exe

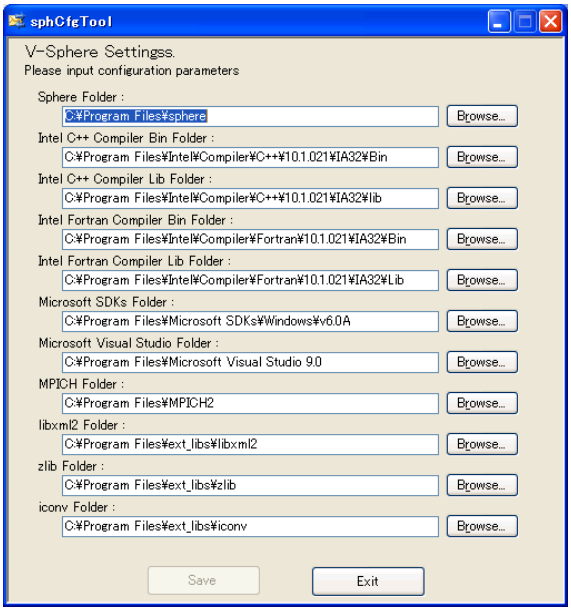


図 5.1　sphCfgTool の設定画面

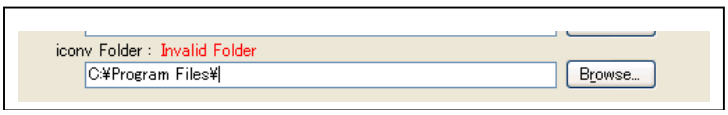


図 5.2　sphCfgTool のパスの設定エラー

# 第 6 章

# Tips

コンパイルなどについて役に立つと思われる情報について記す.

## 6.1　コンパイルエラー

### 6.1.1　VMware 上の Fedora 14 でのコンパイル

VMware 上で Fedrora14 32bit がゲスト OS の場合，V-Sphere Ver. 1.8.4 のコンパイルがうまくいかない場合がある．その場合，以下の手順で成功する事例がある．

```
$ aclocal
$ autoconf
$ automake -a
$ configure [option...]
$ make
```

### 6.1.2　Fedora に mpich2 をインストールした場合のトラブル

Linux OS が Fedora で，並列ライブラリとして mpich2 をインストールした場合，V-Sphere のインストールでリンクエラーとなる場合がある．

その場合，以下のいずれかの手順で成功する事例がある．

**方法 1**

mpich2 のコンパイララッパーを使用する．

mpich2 は、configure 時に指定したコンパイラをラップしたコンパイルコマンドが bin 配下にインストールされる．

mpich2 のインストールディレクトリが/usr/local/mpich2/の場合, /usr/local/mpich2/bin 配下に mpif77, mpif90, mpicc，mpic++ 等のコマンドが格納されている．これらのコマンドは、mpich2 が intel コンパイラで make されていれば、内部的に intel コンパイラがコールされ，かつ，MPI プログラムに必要なライブラリを自動的にリンクしてくれる．

そこで、V-Sphere の configure 時に，

CC=icc
CXX=icpc
FC=ifort
F90=ifort

の代わりに，

CC=/usr/local/mpich2/bin/mpicc
CXX=/usr/local/mpich2/bin/mpic++
FC=/usr/local/mpich2/bin/mpif77
F90=/usr/local/mpich2/bin/mpif90

をそれぞれ指定すると，リンクエラーは解消される．

**方法 2**

不足しているライブラリを強制的にリンクする．

mpich2 の場合、libmpl と libpthread をリンクする必要があるので，これらのリンク指示を Makefile 等に追記することで，リンクエラーを回避する．

以下のファイルを編集する必要がある．

**■V-Sphere コンパイル時**　configure 実行後に，src/utility/sphDataGather/Makefile の以下を修正する．
218、219 行目の「-lmpich」の記述の後に「-lmpl -lpthread」を追加
例）
dataGather_LDADD = -lsphcfg -lmpich -lmpl -lpthread -L/usr/local/mpich2-install/lib -lxml2 -lz -lm
sphDataGather_LDADD = -lsphcfg -lmpich -lmpl -lpthread -L/usr/local/mpich2-install/lib -lxml2 -lz -lm

**■ソルバープロジェクト（CBC）**　project_local_settings の「LIBS=」の行に「-lmpl -lpthread」を追加
（例）
LIBS=-lifport -lifcore -lmpl -lpthread

### 6.1.3　Fedora に OpenMPI をインストールする場合のトラブル

#### OpenMPI のコンフィギュア

Linux OS が Fedora で，並列ライブラリとして OpenMPI のインストールがうまくいかない場合，OpenMPI の configure 時に，

```
--enable-contrib-no-build=vt
```

オプションを付加すると，成功する事例がある．

#### OpenMPI のインストール

Linux OS が Fedora で，並列ライブラリとして OpenMPI をルート権限が必要な場所に

```
$ sudo make install
```

でインストールする場合，

```
icc: command not found
```

というインストールエラーが出る場合がある．

その場合，以下の手順で成功する事例がある．

root ユーザの.bashrc や.bash_profiel に intel コンパイラの PATH などの環境設定を記述する．その後，root ユーザとして，

```
# make install
```

# 第 7 章

# アップデート

本ユーザガイドのアップデート情報について記す.

Version 1.2.2　2011/06/20

- Window モジュール作成を追加.
- 体裁の変更.

Version 1.2.1　2011/06/06

- OpenMPI を Fedora にインストールするときのオプションを Tips に追記.

Version 1.2.0　2011/05/10

- Tips の章を追記.

Version 1.1.9　2011/04/12

- V-Sphere 1.8.4 の機能に対応.

Version 1.1.8　2010/10/31

- コンパイル環境のアップデートに対応.
- インストールと開発環境の構築を分離.

Version 1.1.7　2010/10/09

- 体裁の調整.

Version 1.1.6　2010/07/06

- MPI2 として OpenMPI のインストールを選択.

Version 1.1.5　2010/05/18

- LD_LIBRARY_PATH の export の順序を変更.

Version 1.1.4　2010/03/06

- 「コンフィギュレーション」に用語を統一.

Version 1.1.3　2010/02/10

- RICC のインストールシェルのコンパイルオプションを-O3
- V-Sphere 1.7.7 の出力に更新.
- hyperref を導入.

Version 1.1.2　2010/01/29

- インストールに失敗した場合の記述を修正.
- インストールディレクトリの指定について注意書きを追記.
- 倍精度モジュール時のコメントを追記.

- インストールに失敗する場合の注意事項を修正.
- QUEST の Intel Compiler 11.0 に変更，Intel mpi を追記．シェルのサンプルを変更.
- RICC のインストールシェルに LDFLAGS を追記.

Version 1.1.1　　2010/01/19

- OpenMPI を推奨．mpich2 の記述を削除.
- RICC へのインストールを追記.

Version 1.1.0　　2010/01/05

- OpenMPI を推奨．mpich2 の記述を削除.
- mpirun のコメントを追加.
- 構成変更．sph-cfg.xml について追記.
- ディレクトリ変更に併せて修正.

Version 1.0.0　　2009/03/05

- Version 1.0.0 release

# 参考文献

[1] 小野謙二, 玉木剛. Sphere - 物理シミュレーションのフレームワークと実行環境の開発. 日本計算工学会論文集, No. 20060031, 2006.

[2] 小野謙二, 玉木剛, 野田茂穂, 岩田正子, 重谷隆之. オブジェクト指向並列化クラスライブラリの開発と性能評価. 情報処理学会論文誌：コンピューティングシステム, Vol. 48, No. SIG 8(ACS 18), pp. 44–53, 2007.

[3] Kenji Ono and Tsuyoshi Tamaki. Development of a framework for parallel simulators with various physics and its performance. In Proceeding of International Conference on Parallel Computational Fluid Dynamics, Antalya, Turkey, May 2007. Parallel CFD.

[4] http://www-unix.mcs.anl.gov/mpi/mpich1/.

[5] http://www.mcs.anl.gov/research/projects/mpich2/.

[6] http://www.open-mpi.org/.

# 索引