

WRF/WPS ユーザーガイド

はじめに

これは管理人が WRF/WPS という気象解析予報モデルプログラム及びその初期化プログラムに関して、色々なトラブルに見舞われつつインストール & 使用していった時の覚え書きです。OS は Vine Linux 4.1、コンパイラには Intel C/C++ Compiler 9.1、Intel Fortran Compiler 9.1 を使っています。まあソースからのコンパイル & インストールがほとんどなので、だいたいの UNIX 系 OS なら大丈夫かと思いますが。

注意

このページでは、コマンドラインで実行するコード、ファイルに記述するコードなどが書かれています。コマンドラインで実行するコードについては、#で始まる場合はルート権限での実行とし、\$で始まる場合はユーザ権限で実行します。ファイルに記述するコードは bash の文法に則って記述します。基本的に、root 権限でインストールするプログラムは /usr/local 以下のエントリにインストールしていきますが、後で混乱を避けるために専用ディレクトリを作成し、その中にインストールしていく方式をとっています。ユーザ権限でインストールするプログラムはユーザのホームディレクトリ直下にインストールします。ダウンロードしたファイルに対して作業するときは、/tmp で行います。

目次

WRF/WPS をインストール

- * Intel C/C++ Compiler
- * Intel Fortran Compiler
- * GrADS
- * NetCDF
- * HDF
- * NCAR Graphics
- * NCL
- * WRF
- * WPS
- * WRF2GrADS

WRF/WPS を使う

- * WPS
- * WRF
- * WRF2GrADS

参考情報:旧バージョンのインストール

- * WRFSI (WRF Ver2.1.2 以前に対応)
- * WRF Version 2.1.2

Intel C/C++ Compiler

Intel の非商用ライセンス版コンパイラを取得。Free Non-Commercial Download の Compiler から C/C++ Compiler をダウンロードする。メールアドレスと所在国を入力。そのアドレスにライセンスファイルが送られてくる。ライセンスファイルが送られてきたら、/usr/local 以下にディレクトリを作ってライセンスファイルを置く。

```
# mkdir -p /usr/local/intel/licenses  
# cp /tmp/NCOM_L_CMP_CPP_*.lic /usr/local/intel/licenses
```

次にソースファイルを/tmp 以下に解凍し、作成されたディレクトリに入る。

```
# cd /tmp  
# tar zxvf l_cc_c_9.1.043.tar.gz  
# cd l_cc_c_9.1.043
```

通常ならここでそのままインストールプログラムを実行すればいいが、Vine Linux 3.x の場合はそのまま実行するとインストール途中でエラーが発生しインストールできないので、インストールする際に修正しなければならない箇所がある。data 以下にある install_cc.sh の 1456 行目の記述を修正する。

```
### data/install_cc.sh line:1456  
CMD=`type -pP $c 2>&1` ; # type のオプションを追加
```

これでインストールが可能になる。インストールプログラムを実行。

```
# ./install.sh
```

1→2 の順に選択し、ライセンスファイルのあるアドレスを記述する。
/usr/local/intel/licenses/NCOM_L_CMP_CPP_*.lic 等。標準インストールでデバッガをインストールしても良いが、カスタムインストールで本体だけをインストールしても構わない。インストール先は/usr/local/intel/cc/9.1.043 としておく。インストールが完了したら、ライセンスファイルへのシンボリックリンクを貼っておく。

```
# cd /usr/local/intel/cc/9.1.043/licenses  
# ln -s /usr/local/intel/licenses/NCOM_L_CMP_CPP_*.lic
```

環境設定のパスを通す。

```
### ~/.bashrc  
  
### Intel C/C++ Compiler 9.1.043  
. /usr/local/intel/cc/9.1.043/bin/iccvars.sh
```

環境設定を有効にし、テストが成功したら終了。

```
# . ~/.bashrc
```

Intel Fortran Compiler

C/C++コンパイラ同様、Free Non-Commercial Download から Intel Fortran Compiler をダウンロードする。質問内容は C/C++と全く同じ。メールアドレスと国を記入し、ライセンスファイルを貰う。ライセンスファイルは同じ場所に置いておく。

```
# cp /tmp/NCOM_L_CMP_FOR_*.lic /usr/local/intel/licenses
```

ソースファイルを/tmp 以下に解凍し、作成されたディレクトリに入る。

```
# cd /tmp
# tar zxvf l_fc_c_9.1.037.tar.gz
# cd l_fc_c_9.1.037
```

Fortran コンパイラのインストールでも C/C++コンパイラと同様のインストールエラーが発生する。修正箇所は全く同じ。data 以下の install_fc.sh の 1456 行目を修正する。

```
### data/install_fc.sh line:1456
CMD=`type -pP $c 2>&1` ; # type のオプションを追加
```

インストールプログラムを実行。

```
# ./install.sh
```

1→2 の順に選択し、ライセンスファイルのあるアドレスを記述する。

/usr/local/intel/licenses/NCOM_L_CMP_FOR_*.lic 等。標準インストールでデバッガをインストールしても良いが、カスタムインストールで本体だけをインストールしても構わない。インストール先は/usr/local/intel/fc/9.1.037 としておく。インストールが完了したら、ライセンスファイルへのシンボリックリンクを貼っておく。

```
# cd /usr/local/intel/fc/9.1.037/licenses
# ln -s /usr/local/intel/licenses/NCOM_L_CMP_FOR_*.lic
```

環境設定のパスを通す。

```
### ~/.bashrc

### Intel Fortran Compiler 9.1.037
. /usr/local/intel/fc/9.1.037/bin/ifortvars.sh
```

環境設定を有効にし、テストが成功したら終了。

```
# . ~/.bashrc
```

GrADS

正式名称は Grid Analysis and Display System。解析データの可視化ソフトである。インストールはコンパイル済みのものとソースファイルからのコンパイルがあるが、ここではコンパイル済みのものを選ぶ。

Downloading GrADS Software から各システムの最新バージョンのコンパイル済み Full Distribution を選ぶ。Vine Linux の場合は Intel/LINUX、linuxRHE3 をダウンロード。インストールはこれを展開させるだけで良い。

```
# cd /tmp
# tar zxvf grads-1.9b4-linuxRHE3.tar.gz -C /usr/local
```

次にスクリプトファイルをダウンロード、インストールする。wget を使用し、FTP サイトからデータを取得する。

```
# cd /usr/local/grads-1.9b4
# mkdir scripts
# cd scripts
# wget ftp://grads.iges.org/grads/scripts/*
```

スクリプトファイルのインストールが完了したら、環境設定を記述しておく。

```
### ~/.bashrc

### GrADS 1.9b4
export GRADS="/usr/local/grads-1.9b4"
export PATH="${GRADS}/bin:${PATH}"
export GASCRP="${GRADS}/scripts"
export GADDIR="${GRADS}/data"
```

環境設定を有効にしたら終了。

```
# . ~/.bashrc
```

NetCDF

NetCDF は Network Common Data Form の略である。WRF その他のコンパイルにはこのライブラリが必須となるので、早めにインストールしておく。NetCDF Downloads にアクセスし、最新バージョンをダウンロードする。解凍後、ソースディレクトリへ移動。

```
# cd /tmp
# tar zxvf netcdf.tar.gz
# cd netcdf-3.6.1/src
```

ソースをコンパイルする。netcdf_install.sh というインストール用スクリプトを記述する。

```
#!/bin/sh

export CC=icc
export CXX=icpc
export FC=ifort
export F90=ifort
export CFLAGS="-O2"
export FFLAGS="-O -mp"
export CPPFLAGS="-DNDEBUG -DpgiFortran"

./configure --prefix=/usr/local/netcdf-3.6.1
make test
make install
```

インストール先のディレクトリを作成し、インストール用スクリプトを実行する。

```
# mkdir /usr/local/netcdf-3.6.1
# sh netcdf_install.sh
```

インストール完了後、環境設定を記述する。

```
### ~/.bashrc

### NetCDF 3.6.1
export NETCDF="/usr/local/netcdf-3.6.1"
export NCHOME="${NETCDF}"
export NETCDFHOME="${NETCDF}"
export PATH="${NETCDF}/bin:${PATH}"
export LD_LIBRARY_PATH="${NETCDF}/lib:${LD_LIBRARY_PATH}"
export MANPATH="${NETCDF}/man:${MANPATH}"
```

環境設定後、設定を有効にする。

```
# . ~/.bashrc
```

HDF

HDF は Hierarchical Data Format という NetCDF と同様データフォーマットライブラリの一つである。次にインストールする NCAR Graphics のコンパイル時には予めインストールしておく必要がある。HDF4 と HDF5 があるが、NCARG 4.4.1 の時点では HDF4 のライブラリを必要とする。

はじめに、HDF をインストールするために構文解析器生成プログラムをインストールする必要がある。byacc はパブリックドメイン yacc 構文解析器生成プログラム、bison は GNU 汎用構文解析器生成プログラム、flex はスキャナ(テキストパターン認識器)生成ツールである。各システムで若干名称が違う可能性があるので、該当する構文解析器系のプログラムを確認すること。

他、画像関係の開発用ライブラリも導入する必要がある。これらも適宜確認、インストールする。

```
# apt-get install byacc bison flex
# apt-get install libjpeg libjpeg-devel
# apt-get install zlib zlib-devel
```

HDF Home Page から HDF4 のダウンロードページに行き、Linux 用のソースファイルをダウンロードする。解凍後、ソースディレクトリに移る。

```
# cd /tmp
# tar zxvf HDF4.2r1.tar.gz
# cd HDF4.2r1
```

インストール用スクリプトを記述する。

```
#!/bin/sh

export CC=icc
export CXX=icpc
export F77=ifort
export FC=ifort

./configure --prefix=/usr/local/hdf-4.2r1
make
make install
```

インストール先ディレクトリを作成し、スクリプトを実行する。

```
# mkdir /usr/local/hdf-4.2r1
# sh hdf_install.sh
```

インストール後、bin 以下に nc* 実行ファイルが置かれるが、これは netcdf のインストールで既にインストール済みである。パスの通し方によっては後々 nc* を実行したときに netcdf のものではなくこちらが実行されてしまい、上手く動かなくなる。old ディレクトリを作成し、実行ファイルを移しておく。

```
# cd /usr/local/hdf-4.2r1
# mkdir old
# mv bin/nc* old
```

また、次の ncarg インストール時に HDF のインクルードファイルの書き換えが必要になる。これをバックアップし、修正しておく。

```
# cp include/hdfi.h old

### hdfi.h line:928-931
--- from
#include <sys/types.h>
#include <sys/stat.h>
#include <io.h>
#include <conio.h>
--- to
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/io.h>
#include <ncurses.h>
--- End ---
```

次に環境設定を行う。

```
### ~/.bashrc

### HDF 4.2r1
export HDF="/usr/local/hdf-4.2r1"
export HDFINC="${HDF}/include"
export PATH="${HDF}/bin:${PATH}"
export LD_LIBRARY_PATH="${HDF}/lib:${LD_LIBRARY_PATH}"
```

設定後、設定を有効にする。

```
# . ~/.bashrc
```

NCAR Graphics

NCAR Graphics のインストール。まず、HDF インクルードファイルにあった ncurses がインストールされている必要がある。

```
# apt-get install ncurses ncurses-devel
```

Download NCAR Graphics にアクセスし、ページ中程の「I have read and hereby accept the GNU General Public License.」にチェックする。その他は記入しなくても構わない。ページ下の「Submit」でダウンロードページに移るので、そこから最新のソースファイルを取得する。2006 年 10 月現在は 4.4.1。解凍後、ソースディレクトリに移る。

```
# cd /tmp
# tar zxvf ncarg-4.4.1.src.tar.gz
# cd ncarg-4.4.1
```

インストールファイルを修正する必要があるので、適宜修正。

```
### ./ncarview/src/lib/libncarg_ras/hdf.c line:67-68
--- from
#include <hdf/hdf.h>
#include <hdf/dfgr.h>
--- to
#include <hdf.h>
#include <dfgr.h>
--- End ---
```

./config/LINUX.INTEL の Fortran コマンド ifc は古いコマンド名。現在の ifort に修正しておく。修正後、LINUX ファイルにコピー。

```
### ./config/LINUX.INTEL line:31
#define FCompiler ifort -mp -Vaxlib
```

```
# cp ./config/LINUX.INTEL ./config/LINUX
```

インストール先ディレクトリを作成後、設定用スクリプトを実行する。インストール先ディレクトリ指定は /usr/local/ncarg-4.4.1/* とする。HDF サポートは Yes にし、ライブラリのサーチパスは「/usr/X11R6/lib /usr/local/hdf-4.2r1/lib」「/usr/X11R6/include /usr/local/hdf-4.2r1/include」とする。設定後、コンパイル実行。

```
# mkdir /usr/local/ncarg-4.4.1
# ./Configure
# make Everything >& make-output
```

次に高解像度海岸線データをインストールする。Rainer Feistel にアクセスし、ページ中程にある rangs*.zip、gshhs*.zip を全てダウンロードする。インストール先ディレクトリを作成し、インストール。

```
# mkdir /usr/local/ncarg-4.4.1/rangs
# cd /tmp
# for i in 0 1 2 3 4; \
```

```
> do unzip rangs\($i\).zip; \  
> unzip gshhs\($i\).zip; \  
> done; \  
> mv *.rim *.cat *.cel /usr/local/ncarg-4.4.1/rangs
```

インストール後、環境設定を行う。

```
### ~/.bashrc  
  
### NCARG 4.4.1  
export NCARG="/usr/local/ncarg-4.4.1"  
export NCARG_ROOT="${NCARG}"  
export NCARG_BIN="${NCARG}/bin"  
export NCARG_LIB="${NCARG}/lib"  
export NCARG_INCLUDE="${NCARG}/include"  
export NCARG_RANGS="${NCARG}/rangs"  
export PATH="${NCARG}/bin:${PATH}"  
export LD_LIBRARY_PATH="${NCARG}/lib:${LD_LIBRARY_PATH}"  
export MANPATH="${NCARG}/man:${MANPATH}"
```

設定後、設定を有効にする。

```
# . ~/.bashrc
```

NCL

NCL は NCAR Command Language の略。EarthSystem Grid にアクセスし、「Scientific Data Processing and Visualization Software」→「NCAR Command Language (NCL)」→「NCL 4.2.0.a033 Release (latest)」の順にアクセスする。ログイン画面が現れるので、ID とパスワードを入力する。

新規ユーザーの場合：

1. 「request」をクリックして新アカウントを作成する。
2. アカウント申請ページでは「ESG Contact Person:」以外の全項目に記入。「Request」をクリック。
3. website から申請確認メールが届く。メールの指示に従い、メール内の URL にアクセス。
4. 登録完了までしばらく待つ（一晩ほどかかる）。登録完了メールが来たらメールの指示に従う。
5. ログインページにて ID とパスワードを入力。ログインページ下の「I Accept」をクリック。

「NCL 4.2.0.a033 binaries (not OPeNDAP-enabled)」→「NCL 4.2.0.a033 32-bit binary for i686 chips for LINUX (using GNU)」の順にクリックし、ファイルをダウンロードする。ダウンロードしたファイルは/tmp に移動して、解凍する。

```
# cd /tmp
# tar zxvf ncl-4.2.0.a033.Linux_i686.tar.gz -C /usr/local/ncarg-4.4.1
```

以上でインストールが完了。環境設定を記述する。

```
### ~/.bashrc

### NCL 4.2.0
export NCLCOMMAND="${NCARG}/bin/idt"
export NCL_COMMAND="${NCARG}/bin/ncl"
```

設定後、設定を有効にする。

```
# . ~/.bashrc
```

以下、テスト項目。これが正常に動けば、インストールは問題ない。

```
--- NCARG Test ---
# ncargex cpex08
# idt cpex08.ncgm
# ctrans -d X11 cpex08.ncgm

--- NCL Test ---
# ng4ex gsun01n
# ncl gsun01n.ncl
```

WRF

Weather Research and Forecasting Model(気象解析予報モデル)。気象シミュレーション解析プログラムの一種。
WRF Download から WRF model tar file をダウンロード。

WRF はユーザー権限でインストールする。mkdir でホームディレクトリ下に WRF 用のディレクトリを作成する。

```
$ mkdir ~/WRF
$ cd ~/WRF
$ tar zxvf /tmp/WRFV2.2.TAR.gz
$ cd WRFV2
```

arch/configure.defaults の先頭に、icc/ifort の設定を追加する(ページの構成上、折り返し行が存在することに注意)。

```
#####
#ARCH   PC Linux i486 i586 i686, ifort+icc compiler (Single-threaded, no nesting)
OMP     =
OMPCPP  =
FC      =   ifort
CC      =   icc
SCC     =   $(CC)
SFC     =   $(FC)
RWORDSIZE = $(NATIVE_RWORDSIZE)
RSIZEBITS = `expr $(RWORDSIZE) \* 8`
PROMOTION = -real_size $(RSIZEBITS)
FCDEBUG =   # -g
FCBASEOPTS = -w -FR -cm -I. -Vaxlib -convert big_endian -mp
FCOPTIM  =   -O2
FCFLAGS  =   $(FCOPTIM) $(FCBASEOPTS) $(OMP)
CFLAGS   =   -w
# machine-specific flags needed to link in ESMF library (C++ run-time-library, etc.)
ESMF_LIB_FLAGS =
ESMF_IO_LIB     =   ESMFIOLIB
ESMF_IO_LIB_EXT =   ESMFIOEXTLIB
INCLUDE_MODULES =   -module ../main -I../external/io_netCDF -I../external/io_int
-I../external/esmf_time_f90 \
-I../external -I../frame -I../share -I../phys -I../inc -I../chem
ARCHFLAGS = -DLIMIT_ARGS -DIWORDSIZE=4 -DDWORDSIZE=8 -DRWORDSIZE=$(
(RWORDSIZE) \
-DLWORDSIZE=4 CONFIGURE_NETCDF_FLAG -DGRIB1
CONFIGURE_GRIB2_FLAG #-DIFORT_KLUDGE
LD      =   $(FC)
LDFLAGS =   $(FCFLAGS) $(OMP) CONFIGURE_LDFLAGS
ENVCOMPDEFS =   CONFIGURE_COMPILEFLAGS
CPP     =   ifort -E -free
POUND_DEF = $(OMPCPP) $(COREDEFS) -DNONSTANDARD_SYSTEM \
-DCONFIG_BUF_LEN=$(CONFIG_BUF_LEN) -DMAX_DOMAINS_F=$(
(MAX_DOMAINS)
CPPFLAGS = -I$(LIBINCLUDE) -C -P $(ARCHFLAGS) $(ENVCOMPDEFS) $
(POUND_DEF)
PERL     =   CONFIGURE_PERL_PATH
REGISTRY =   Registry
```

```

LIB          =      CONFIGURE_NETCDF_LIB_PATH ../frame/module_internal_header_util.o ../
frame/pack_utils.o \
              -L../external/esmf_time_f90 -lesmf_time \
              -L../external/io_grib1 -lio_grib1 \
              CONFIGURE_GRIB2_LIB \
              -L../external/io_grib_share -lio_grib_share
AR           =      ar ru
M4           =      m4 -B14000
RANLIB       =      ranlib
NETCDFPATH  =      CONFIGURE_NETCDF_PATH
CC_TOOLS    =      $(CC)

externals : wrf_ioapi_includes CONFIGURE_WRFIO_NF wrfio_grib_share wrfio_grib1
CONFIGURE_WRFIO_GRIB2 wrfio_int module_dm.F esmf_time

module_dm.F :
    ( /bin/cp module_dm_warning module_dm.F ; cat module_dm_stubs.F >> module_dm.F )

wrfio_nf :
    ( cd ../external/io_netcdf ; \
      make NETCDFPATH=CONFIGURE_NETCDF_PATH RANLIB="$(RANLIB)" CPP="$
(CPP)" FC="$(SFC) $(PROMOTION) -FR -I. -w" )

wrfio_grib_share :
    ( cd ../external/io_grib_share ; \
      make CC="$(SCC)" CFLAGS="$(CFLAGS)" RM="$(RM)" RANLIB="$(RANLIB)" CPP="$
(CPP)" \
      FC="$(SFC) $(PROMOTION) -I. $(FCDEBUG) $(FCFLAGS) -w" archive)

wrfio_grib1 :
    ( cd ../external/io_grib1 ; \
      make CC="$(SCC)" CFLAGS="$(CFLAGS)" RANLIB="$(RANLIB)" CPP="$(CPP)" FC="$
(SFC) $(PROMOTION) -I. $(FCDEBUG) $(FCFLAGS) -w" archive)

wrfio_grib2 :
    ( cd ../external/io_grib2 ; \
      make CC="$(SCC)" CFLAGS="$(CFLAGS) CONFIGURE_GRIB2_INC" RM="$(RM)"
RANLIB="$(RANLIB)" CPP="$(CPP)" FC="$(SFC) $(PROMOTION) -I. $(FCDEBUG) $
(FCFLAGS) -w" FIXED="-fixed" archive)

wrfio_int :
    ( cd ../external/io_int ; \
      make CC="$(CC)" RANLIB="$(RANLIB)" CPP="$(CPP)" SFC="$(SFC) $(PROMOTION) $
(FCDEBUG) $(FCBASEOPTS)" FC="$(FC) $(PROMOTION) $(FCDEBUG) $(FCFLAGS) -w" all )

esmf_time :
    ( cd ../external/esmf_time_f90 ; \
      make FC="$(FC) $(PROMOTION) $(FCDEBUG) $(FCFLAGS)" RANLIB="$(RANLIB)"
CPP="$(CPP) -I../inc -I. $(POUND_DEF)" )

# compile these without high optimization to speed compile
solve_interface.o : solve_interface.F
shift_domain_em.o : shift_domain_em.F
module_io_mm5.o : module_io_mm5.F

```

```

module_si_io.o : module_si_io.F
module_io_wrf.o : module_io_wrf.F
module_domain.o : module_domain.F
module_start.o : module_start.F
module_initialize.o : module_initialize.F
module_initialize_b_wave.o : module_initialize_b_wave.F
module_initialize_hill2d_x.o : module_initialize_hill2d_x.F
module_initialize_quarter_ss.o : module_initialize_quarter_ss.F
module_initialize_squall2d_x.o : module_initialize_squall2d_x.F
module_initialize_squall2d_y.o : module_initialize_squall2d_y.F
module_initialize_real.o : module_initialize_real.F
start_domain.o : start_domain.F
wrf_bdyin.o : wrf_bdyin.F
wrf_bdyout.o : wrf_bdyout.F
wrf_histin.o : wrf_histin.F
wrf_histout.o : wrf_histout.F
wrf_inputin.o : wrf_inputin.F
wrf_inputout.o : wrf_inputout.F
wrf_restartin.o : wrf_restartin.F
wrf_restartout.o : wrf_restartout.F

wrf_bdyin.o wrf_bdyout.o \
wrf_histin.o wrf_histout.o \
wrf_inputin.o wrf_inputout.o \
wrf_restartin.o wrf_restartout.o \
solve_interface.o \
shift_domain_em.o \
module_io_mm5.o module_si_io.o module_io_wrf.o module_domain.o \
module_start.o module_initialize.o module_initialize_b_wave.o \
module_initialize_hill2d_x.o module_initialize_quarter_ss.o \
module_initialize_squall2d_x.o module_initialize_squall2d_y.o \
module_initialize_real.o module_dm.o start_domain.o :
$(RM) $@
$(SED_FTN) *.F > *.b
$(CPP) -I./inc $(CPPFLAGS) *.b > *.f90
$(RM) *.b
$(FC) -c $(PROMOTION) $(FCBASEOPTS) $(MODULE_DIRS) *.f90

module_dm.o : module_dm.F
module_configure.o : module_configure.F

module_configure.o \
module_dm.o :
$(RM) $@
$(SED_FTN) *.F > *.b
$(CPP) -I./inc $(CPPFLAGS) *.b > *.f90
$(RM) *.b
$(FC) -c $(PROMOTION) $(FCBASEOPTS) $(MODULE_DIRS) -g -O0 *.f90

```

「AMD x86_64 Intel xeon i686 ia32 Xeon Linux, ifort compiler (single-threaded, no nesting)」の設定を基にしている。ifort コンパイラでは、DIFORT_KLUDGE のフラグがあると module_configure.f90 のコンパイル時にメモリを消費しきってしまい、フリーズする。そのため、このフラグは使用しないようにコメントアウトする必要がある。他、C コンパイラを icc、Fortran コンパイラを ifort、CPP を ifort に変更。

設定の変更が終了したら configure を実行する。設定は 1 の追加した設定を使用する。configure が終わったらコンパイル。

```
$. /configure  
$. /compile em_real
```

run ディレクトリに ndown.exe、real.exe、wrf.exe が作成されればコンパイル成功。

WPS

WRF Preprocessing System の略。WRFSI に代わり、WRF Version2.2 から WPS が前処理プログラムとなる。WRFSI から大きく変更された点は、WPS の前に WRF をインストールする必要があること、GUI だけでなく CUI でも処理が可能になったこと等。ここでは簡単のために、WPS の GUI をインストールしない。

WRF Download のページから WPS をダウンロードする。また地理データとして geog.tar.gz もダウンロードすること。WRFSI のものと若干異なるので、WPS 用に新しくインストールする必要がある。

```
$ cd ~/WRF
$ tar zxvf /tmp/WPSV2.2.TAR.gz
$ tar zxvf /tmp/geog.tar.gz
$ mv geog GEOG
$ cd WPS
```

WRF 同様、arch ディレクトリに configure.defaults があるので該当箇所を修正。

```
### arch/configure.defaults line:253-265
CC          =      icc
SCC         =      icc
CPP         =      ifort -E -free -C -P
```

他、zlib 等 GRIB2 を使うために画像関係のライブラリを導入する必要がある。Vine Linux なら apt-get で簡単にインストールできる。

```
# apt-get install zlib zlib-devel (HDF インストール時に導入済み)
# apt-get install libpng libpng-devel
# apt-get install jasper jasper-devel
```

configure で設定。設定モードは修正を施した 6. を選択する。設定後、コンパイルを実行。コンパイルには、WPS のルートディレクトリと同階層に "WRFV2" というディレクトリが存在し、その中に WRF がインストールされている必要がある。

```
$ ./configure
$ ./compile
```

geogrid.exe、ungrib.exe、metgrid.exe の 3 つが出力されればコンパイル完了。

WRF2GrADS

WRF で出力されたファイルを GrADS で可視化する。この出力がうまくいけば、一通りの解析とその表示が可能になる。WRF Download の「WRF Post-Processing Software」から「Convert WRF model output in netCDF to GrADS format」にアクセスし、WRF2GrADS をダウンロードする。解凍後、ディレクトリへ移動。

```
$ cd ~/WRF
$ tar zxvf /tmp/wrf2grads.tar.gz
$ cd WRF2GrADS
```

インストールの設定を行う。Makefile の「linux flag (INTEL)」の箇所を修正。コメントアウトを解除し、ライブラリファイル、インクルードファイルへのパスを修正する。CPP やフラグも修正しておく。

```
### Makefile line:29-34
LIBNETCDF = -L/usr/local/netcdf-3.6.1/lib -lnetcdf -lm
INCLUDE = -I/usr/local/netcdf-3.6.1/include -I./
FC = ifort
FCFLAGS = -C -FR
CPP = ifort -E -free
CPPFLAGS = -I. -C -DRECL1
```

また、このままコンパイルすると「forrtl: severe (193): Run-Time Check Failure. The variable 'module_wrf_to_grads_util_mp_time_calc_\$HOUR1' is being used without being defined」という実行時エラーを起こす。これは初期化されていない変数を使用しているというエラー。詳細は「インテル(R) Visual Fortran コンパイラ 9.1 Windows* 版 リリースノート」の変更点のうちの該当箇所を参照(Linux 版のリリースノートでは 9.0、9.1 ともにそのような変更点は記載されていない)。これを回避するために、module_wrf_to_grads_util.F の 1066-1068 行目を修正する。

```
### module_wrf_to_grads_util.F line:1066-1068
--- from ---
hour1=hours
mins1=minutes
elseif ( it == 2) then
--- to ---
elseif ( it == 2) then
hour1=hours
mins1=minutes
```

アルゴリズムやインデントの状態を考慮すると、書き順ミスによる単純なバグ? のように見える。

修正が終わったらコンパイルを実行しても良いが、他のソースファイルまで実行可能になっているので色分けをしているときに見づらくなる。属性を変更してからコンパイルしても支障はない。

```
$ chmod 644 *
$ make
```

wrf_to_grads 実行ファイルが作成されればコンパイル終了。

WPS

WPSを使うにはFNLデータを必要とする。WRF Downloadにある「WRF Preprocessing System test data」をダウンロードし、適当なディレクトリを作成して解凍。

```
$ cd ~/WRF
$ mkdir -p FNL/sample
$ tar zxvf /tmp/avn_data.tar.gz -C FNL/sample
$ cd WPS
```

WPSは「geogrid.exe」→「ungrib.exe」→「metgrid.exe」の順でプログラムを走らせる。これらのプログラムの設定には、namelist.wpsを一貫して使用する。

まず geogrid.exe を走らせる。namelist.wps を以下のように編集する。

```
### namelist.wps

&share
wrf_core = 'ARW',
max_dom = 1,
start_date = '2000-01-24_12:00:00',
end_date = '2000-01-25_00:00:00',
interval_seconds = 21600,
io_form_geogrid = 2,
/

&geogrid
parent_id      = 1,
parent_grid_ratio = 1,
i_parent_start = 1,
j_parent_start = 1,
e_we          = 74,
e_sn          = 61,
geog_data_res = '10m'
dx = 30000,
dy = 30000,
map_proj = 'lambert',
ref_lat = 34.83,
ref_lon = -81.03,
truelat1 = 30.0,
truelat2 = 60.0,
stand_lon = -90.0,
geog_data_path = './GEOG'
/
```

ファイルの編集が終了したら、geogrid.exe を走らせる。

```
$ ./geogrid.exe
```

これで地理データの初期化が完了する。次に ungrib.exe の設定を行う。

```
### namelist.wps
```

```
&ungrib  
  out_format = 'WPS',  
  prefix = 'NNRP',  
/
```

FNL データにアクセスするために、シンボリックリンクを設定する。シンボリックリンクを貼るための C シェルスクリプトが link_grib.csh として提供されているのでこれを利用する。

```
$ ./link_grib.csh ../FNL/sample/*
```

処理する変数テーブルに対してシンボリックリンクを貼る。ここでは NNRP を使用する。

```
$ ln -sf ungrib/Variable_Tables/Vtable.NNRP Vtable
```

設定が終了したら、ungrib.exe を走らせる。

```
$ ./ungrib.exe
```

さらに NNRPSPFC の処理を行う。namelist.wps を編集し、Vtable に対するシンボリックリンクを張り替えて ungrib.exe を実行する。

```
### namelist.wps
```

```
&ungrib  
  out_format = 'WPS',  
  prefix = 'NNRPSFC',  
/
```

```
$ ln -sf ungrib/Variable_Tables/Vtable.NNRPSFC Vtable
```

```
$ ./ungrib.exe
```

最後に metgrid.exe を使用して WRF の初期化データを出力する。namelist.wps を以下のように編集する。

```
### namelist.wps
```

```
&metgrid  
  fg_name = 'NNRP','NNRPSFC',  
  io_form_metgrid = 2,  
/
```

編集が終了したら、metgrid.exe を走らせる。

```
$. /metgrid.exe
```

出力に成功したら、met_em.*.nc というファイル群が作成される。WRF ではこのファイル群を使用する。

WRF

WRF は run ディレクトリでプログラムを実行する。ディレクトリに移動し、WPS で作成したファイルをこのディレクトリに置く。

```
$ cd ~/WRF/WRFV2/run
$ mv ~/WRF/WPS/met_em* .
```

WRF の設定は namelist.input で行う。namelist.input に関する詳しい情報は、README.namelist を参照すること。以下、設定ファイルの情報を記す。

```
### namelist.input

&time_control
run_days           = 0,
run_hours          = 12,
run_minutes        = 0,
run_seconds        = 0,
start_year         = 2000, 2001, 2001,
start_month        = 01, 06, 06,
start_day          = 24, 11, 11,
start_hour         = 12, 12, 12,
start_minute       = 00, 00, 00,
start_second       = 00, 00, 00,
end_year           = 2000, 2001, 2001,
end_month          = 01, 06, 06,
end_day            = 25, 12, 12,
end_hour           = 00, 12, 12,
end_minute         = 00, 00, 00,
end_second         = 00, 00, 00,
interval_seconds   = 21600
input_from_file    = .true.,.false.,.false.,
history_interval   = 180, 60, 60,
frames_per_outfile = 1000, 1000, 1000,
restart            = .false.,
restart_interval    = 5000,
io_form_history     = 2
io_form_restart    = 2
io_form_input      = 2
io_form_boundary   = 2
debug_level        = 0
/

&domains
time_step          = 180,
time_step_fract_num = 0,
time_step_fract_den = 1,
max_dom            = 1,
s_we               = 1, 1, 1,
e_we               = 74, 112, 94,
s_sn               = 1, 1, 1,
```

```

e_sn           = 61, 97, 91,
s_vert        = 1, 1, 1,
e_vert        = 27, 28, 28,
num_metgrid_levels = 27
dx            = 30000, 3333, 1111,
dy            = 30000, 3333, 1111,
grid_id       = 1, 2, 3,
parent_id     = 0, 1, 2,
i_parent_start = 0, 30, 30,
j_parent_start = 0, 20, 30,
parent_grid_ratio = 1, 3, 3,
parent_time_step_ratio = 1, 3, 3,
feedback      = 1,
smooth_option = 0
/

&physics
mp_physics    = 3, 3, 3,
ra_lw_physics = 1, 1, 1,
ra_sw_physics = 1, 1, 1,
radt          = 30, 30, 30,
sf_sfclay_physics = 1, 1, 1,
sf_surface_physics = 1, 1, 1,
bl_pbl_physics = 1, 1, 1,
bldt         = 0, 0, 0,
cu_physics    = 1, 1, 0,
cudt         = 5, 5, 5,
isfflx       = 1,
ifsnow       = 0,
icloud       = 1,
surface_input_source = 1,
num_soil_layers = 5,
ucmcall      = 0,
mp_zero_out  = 0,
maxiens      = 1,
maxens       = 3,
maxens2      = 3,
maxens3      = 16,
ensdim       = 144,
/

&fdda
/

&dynamics
w_damping     = 0,
diff_opt      = 1,
km_opt        = 4,
diff_6th_opt  = 0,
diff_6th_factor = 0.12,
base_temp     = 290.

```

```

damp_opt          = 0,
zdamp            = 5000., 5000., 5000.,
dampcoef         = 0.01, 0.01, 0.01
khdif            = 0, 0, 0,
kvdif            = 0, 0, 0,
non_hydrostatic  = .true., .true., .true.,
pd_moist         = .false., .false., .false.,
pd_scalar        = .false., .false., .false.,
/

&bdy_control
spec_bdy_width   = 5,
spec_zone        = 1,
relax_zone       = 4,
specified        = .true., .false., .false.,
nested           = .false., .true., .true.,
/

&grib2
/

&namelist_quilt
nio_tasks_per_group = 0,
nio_groups = 1,
/

```

設定が完了したら、初期値・境界値データを作成する。Vine Linux ではスタックサイズに 8192KB 制限があるので、real.exe を実行する前にスタックサイズを無制限にしておく。

```

$ ulimit -s unlimited
$ ./real.exe

```

初期値・境界値データが wrfbdy、wrfinput というファイルで作成される。この後、wrf.exe を実行してシミュレーションを行う。

```

$ ulimit -s unlimited
$ ./wrf.exe

```

計算範囲やタイムステップによって実行時間は大きく変化する。シミュレーションが終了すると、wrfout ファイルが作成される。

WRF2GrADS

WRF で作成されたファイルは NetCDF のファイル形式をとっている。この出力ファイルを GrADS で表示できるように変換する。WRF で作成したファイルを WRF2GrADS ディレクトリに移動する。

```
$ cd ~/WRF/WRF2GrADS
$ mv ~/WRF/WRFV2/run/wrfout* .
```

GrADS 変換プログラムを走らせるために、control_file を編集する。

```
### control_file

-2          ! number of times to put in GrADS file, negative means ignore the times
2000-01-24_12:00:00
2000-01-24_18:00:00
2000-01-25_00:00:00
end_of_time_list

          ! 3D variable list for GrADS file
          ! indent one space to skip
U         ! U Component of wind
V         ! V Component of wind
UMET      ! U Component of wind - rotated (diagnostic)
VMET      ! V Component of wind - rotated (diagnostic)
W         ! W Component of wind
THETA     ! Theta
TK        ! Temperature in K
TC        ! Temperature in C
TKE       ! TURBULENCE KINETIC ENERGY
P         ! Pressure (hPa)
Z         ! Height (m)
QVAPOR    ! Vapor
QCLOUD    ! Cloud Water
QRAIN     ! Rain Water
QICE      ! -
QSNOW     ! -
QGRAUP    ! -
TKE_MYJ   ! TKE FROM MELLOR-YAMADA-JANJIC
PB        ! BASE STATE PRESSURE AT HALF LEVEL
TD        ! Dewpoint Temperature (diagnostic)
RH        ! Relative Humidity (diagnostic)

          ! Soil variables
TSLB      ! SOIL TEMPERATURE
SMOIS     ! SOIL MOISTURE

          ! below a list of fields from SI static and input files
LANDUSEF  ! -
SOILCTOP  ! -
SOILCBOT  ! -
SPECHUMD  ! -
GREEN12M  ! -
ALBDO12M  ! -
end_of_3dvar_list
ACSNOM    ! ACCUMULATED MELTED SNOW
ACSNOW    ! ACCUMULATED SNOW
```

AKHS ! SFC EXCH COEFF FOR HEAT
 AKMS ! SFC EXCH COEFF FOR MOMENTUM
 CANWAT ! CANOPY WATER
 GLW ! DOWNWARD LONG WAVE FLUX AT GROUND SURFACE
 GSW ! DOWNWARD SHORT WAVE FLUX AT GROUND SURFACE
 HFX ! UPWARD HEAT FLUX AT THE SURFACE
 HGT ! Terrain Height
 IVGTYP ! VEGETATION TYPE
 ISLTYP ! SOIL TYPE
 LU_INDEX ! LAND USE CATEGORY
 MAPFAC_M ! Map scale factor on mass grid
 MU ! Perturbation dry air mass in column
 MUB ! base state dry air mass in column
 MU0 ! initial dry mass in column
 Q2 ! QV at 2 M
 QFX ! UPWARD MOISTURE FLUX AT THE SURFACE
 RAINC ! ACCUMULATED TOTAL CUMULUS PRECIPITATION
 RAINCV ! TIME-STEP CUMULUS PRECIPITATION
 RAINNC ! ACCUMULATED TOTAL GRID SCALE PRECIPITATION
 SFROFF ! SURFACE RUNOFF
 slvl ! sea level pressure
 SMSTAV ! MOISTURE VARIBILITY
 SNOW ! SNOW WATER EQUIVALENT
 SNOWC ! FLAG INDICATING SNOW COVERAGE (1 FOR SNOW COVER)
 SST ! SEA SURFACE TEMPERATURE
 T2 ! TEMP at 2 M
 TH2 ! POT TEMP at 2 M
 TMN ! SOIL TEMPERATURE AT LOWER BOUNDARY
 TSK ! SURFACE SKIN TEMPERATURE
 U10 ! U at 10 M
 U10M ! U at 10 M - rotated
 V10 ! V at 10 M
 V10M ! V at 10 M - rotated
 UDROFF ! UNDERGROUND RUNOFF
 VEGFRA ! VEGETATION FRACTION
 WEASD ! WATER EQUIVALENT OF ACCUMULATED SNOW
 XLAT ! LATITUDE, SOUTH IS NEGATIVE
 XLONG ! LONGITUDE, WEST IS NEGATIVE
 XLAND ! LAND MASK (1 FOR LAND, 2 FOR WATER)
 ! below a list of fields from SI static and input files
 ALBBCK ! -
 LANDMASK ! -
 PMSL ! -
 SLOPECAT ! -
 SHDMAX ! -
 SHDMIN ! -
 SNOALB ! -
 SOILHGT ! -
 ST000010 ! -
 ST010040 ! -
 ST040100 ! -
 ST100200 ! -
 SM000010 ! -
 SM010040 ! -

```

SM040100      ! -
SM100200      ! -
TOPOSTDV      ! -
TOPOSLPX      ! -
TOPOSLPY      ! -
XICE          ! -
end_of_2dvar_list
              ! All list of files to read here
              ! Indent not to read
              ! Full path OK
/DATA/wrfstatic_d01
/DATA/wrfstatic_d02
/DATA/wrf_real_input_em.d01.2000-01-24_12:00:00
/DATA/wrf_real_input_em.d02.2000-01-24_12:00:00
/DATA/real/wrfinput_d01

./wrfout_d01_2000-01-24_12:00:00

/DATA/b_wave/wrfout_d01_0001-01-01_00:00:00
/DATA/grav2d_x/wrfout_d01_0001-01-01_00:00:00
/DATA/hill2d_x/wrfout_d01_0001-01-01_00:00:00
/DATA/quarter_ss/wrfout_d01_0001-01-01_00:00:00
/DATA/squall2d_x/wrfout_d01_0001-01-01_00:00:00
/DATA/squall2d_y/wrfout_d01_0001-01-01_00:00:00
end_of_file_list
              ! Now we check to see what to do with the data
real          ! real (input/output) / ideal / static
1             ! 0=no map background in grads, 1=map background in grads
-1           ! specify grads vertical grid
              ! 0=cartesian,
              ! -1=interp to z from lowest h
              ! 1 list levels (either height in km, or pressure in mb)

1000.0
950.0
900.0
850.0
800.0
750.0
700.0
650.0
600.0
550.0
500.0
450.0
400.0
350.0
300.0
250.0
200.0
150.0
100.0

```

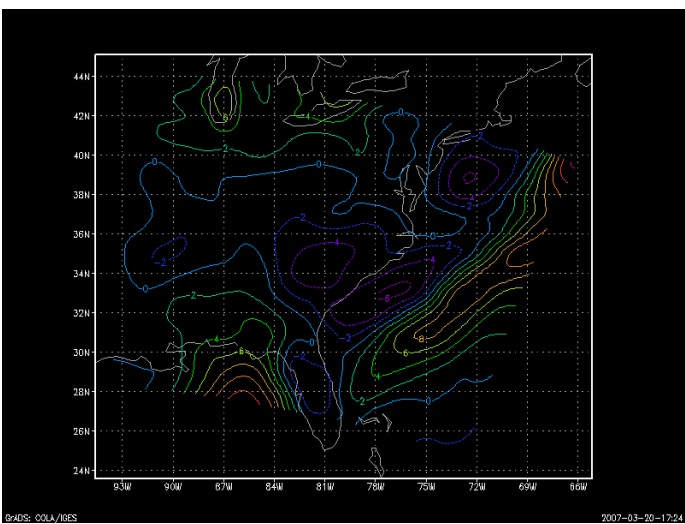
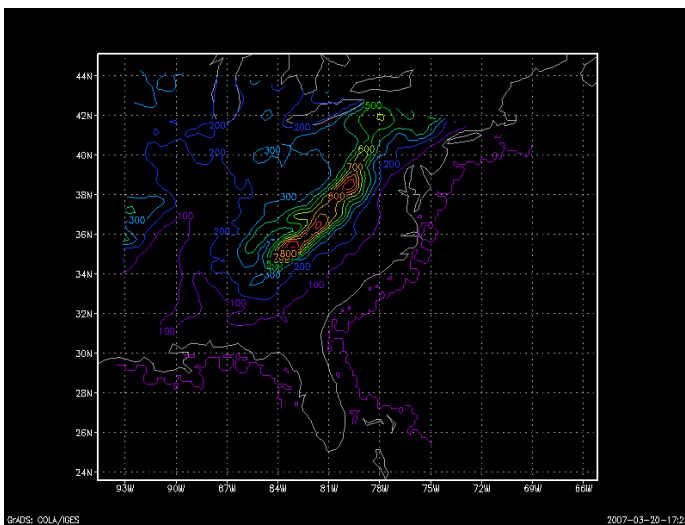
編集が終了したら、変換プログラムを走らせる。

```
$.wrf_to_grads control_file sample
```

sample は出力名となる。出力が完了すると、GrADS 形式の sample.ctl と sample.dat が作成される。GrADS を起動し、出力を確認する。

```
$ gradsnc  
ga-> open sample.ctl  
ga-> q file  
ga-> d hgt  
ga-> c  
ga-> d u10m
```

設定通りに行うと、以下のような画像が出力される。



参考情報:旧バージョンのインストール

WRFSI

※ WRF Version2.2 から WRFSI の代わりに WPS が前処理プログラムとして使われるようになっている。

WRFSI は次にインストールする WRF の初期化処理プログラム (Standard Initialization: SI) である。WRFSI からダウンロードする。研究開発用の ARW 型と、現業用の NMM 型があるが、ここでは ARW を選択。また、ダウンロードページから地理データ (Geog Data) にアクセスするスクリプトファイルもダウンロードしておく。

WRFSI は Perl-Tk を使って GUI を提供している。2006 年 10 月現在の Vine Linux 3.2 の場合、この Perl-Tk のバージョンは 804.026 であるが、このバージョンでは GUI がうまく動かない。バージョン 800.025 が必要になる。このバージョンの Perl-Tk があればアンインストールしておき、独自に Perl-Tk をインストールする。

SourceForge.net から該当バージョンのソースファイルをダウンロード。解凍し、ディレクトリへ移動する。

```
# apt-get remove perl-Tk
# cd /tmp
# tar jxvf perl-Tk-800.025-2-src.tar.bz2
# tar jxvf Tk-800.025.tar.bz2
# cd Tk-800.025
```

Makefile を Perl スクリプトで作成し、インストールする。インストール先は /usr/local/perl-Tk-800.025 とする。作成した Makefile を一部修正してインストール。このときはインストールに gcc を使うので、初期設定のままでもよい。

```
# perl Makefile.PL

### Makefile
PREFIX = /usr/local/perl-Tk-800.025
PERLPREFIX = /usr/local/perl-Tk-800.025
SITEPREFIX = /usr/local/perl-Tk-800.025
VENDORPREFIX = /usr/local/perl-Tk-800.025
```

```
# mkdir /usr/local/perl-Tk-800.025
# make
# make test
# make install
```

これらが完了したら、パスを通すためのシンボリックリンクを作成する。その後、テストがうまくいったら完了。

```
# cd /usr/local/lib
# rm -rf site_perl
# ln -s /usr/local/perl-Tk-800.025/lib/perl5/site_perl
# perl -e 'use Tk; print "\to This is Perl/Tk, version $Tk::VERSION installed in library $Tk::library\n"
```

WRFSI はユーザ権限でインストールする。インストール先はホームディレクトリ直下に WRF ディレクトリを作成し、その中とする。解凍し、ディレクトリへ移動。

```
$ cd ~
```

```
$ mkdir WRF
$ cd WRF
$ tar zxvf /tmp/wrfsti_v2.1.2.tar.gz
$ cd wrfsti
```

WRFSTI のコンパイルには NetCDF のインストールが必須であり、同じコンパイラでコンパイルする必要がある。もしコンパイルが上手く通らない場合は、ちゃんと NetCDF のインストールが同じコンパイラで成功しているかどうか確認すること。ここでは icc/ifort を一貫して使っているので問題はないと思うが。

ソースファイルの記述を修正する。./src/include/makefile_pcintel.inc.in の該当箇所を確認し、修正。

```
### ./src/include/makefile_pcintel.inc.in
CC = icc
NCARGFC = /usr/local/ncarg-4.4.1/bin/ncargf77 # PATH は適宜変更
CPP = ifort -E -free
CPPFLAGS = $(INC) $(DEFS)
```

また、このバージョンでは hinterp.exe のコンパイルエラー問題がある。hinterp.exe をコンパイルする際に、「undefined reference to nc*... (nc*関数が定義 されていない)」という一見して NetCDF のインクルードがうまくいっていないように思えるエラーが発生する。これは NetCDF の問題ではなく、hinterp の Makefile にバグが存在するためである。以下に修正用の Makefile を記述しているので、これを src/hinterp/Makefile と差し替えること。

参考:

- * http://tornado.meso.com/wrf_forum/index.php?showtopic=694
- * <http://cfa-www.harvard.edu/~tkurosu/OnE/WRF/index.html>

```
#dis
#dis  Open Source License/Disclaimer, Forecast Systems Laboratory
#dis  NOAA/OAR/FSL, 325 Broadway Boulder, CO 80305
#dis
#dis  This software is distributed under the Open Source Definition,
#dis  which may be found at http://www.opensource.org/osd.html.
#dis
#dis  In particular, redistribution and use in source and binary forms,
#dis  with or without modification, are permitted provided that the
#dis  following conditions are met:
#dis
#dis  - Redistributions of source code must retain this notice, this
#dis  list of conditions and the following disclaimer.
#dis
#dis  - Redistributions in binary form must provide access to this
#dis  notice, this list of conditions and the following disclaimer, and
#dis  the underlying source code.
#dis
#dis  - All modifications to this software must be clearly documented,
#dis  and are solely the responsibility of the agent making the
#dis  modifications.
#dis
#dis  - If significant modifications or enhancements are made to this
#dis  software, the FSL Software Policy Manager
#dis  (softwaremgr@fsl.noaa.gov) should be notified.
#dis
```

```
#dis THIS SOFTWARE AND ITS DOCUMENTATION ARE IN THE PUBLIC DOMAIN
#dis AND ARE FURNISHED "AS IS." THE AUTHORS, THE UNITED STATES
#dis GOVERNMENT, ITS INSTRUMENTALITIES, OFFICERS, EMPLOYEES, AND
#dis AGENTS MAKE NO WARRANTY, EXPRESS OR IMPLIED, AS TO THE USEFULNESS
#dis OF THE SOFTWARE AND DOCUMENTATION FOR ANY PURPOSE. THEY ASSUME
#dis NO RESPONSIBILITY (1) FOR THE USE OF THE SOFTWARE AND
#dis DOCUMENTATION; OR (2) TO PROVIDE TECHNICAL SUPPORT TO USERS.
#dis
```

```
SRCROOT=./..
```

```
include $(SRCROOT)/src/include/makefile.inc
```

```
RM=rm -f
```

```
.SUFFIXES:          .F .o
```

```
.F.o: $(FOBJS)
      $(RM) $@
      $(FC) -c $(FFLAGS) $(FREE) $(INC) $(MODULELIB) $(LAPSLIB) -c $< -o $@
```

```
EXE= hinterp.exe
```

```
FSRC=      hinterp.F \
           module_domain_info.F \
           module_gridded_data.F \
           module_hinterp_setup.F \
           module_hinterp_gribprep.F \
           proc_make_variable_metadata.F \
           proc_output_variable.F \
           proc_store_global_metadata.F \
           wrf_debug.F
```

```
FOBJS=$(FSRC:.F=.o)
```

```
all:      $(EXE)
```

```
$(EXE):      $(FOBJS)
             $(FC) -o $(EXE) $(LD_FLAGS) $(FOBJS) $(LAPSLIB) $(MODULELIB) $(IOAPI) \
             $(OTHERLIBS)
```

```
hinterp.o:      module_hinterp_setup.o module_domain_info.o module_gridded_data.o \
               module_hinterp_gribprep.o wrf_debug.o
```

```
module_domain_info.o:      module_hinterp_setup.o
```

```
module_hinterp_gribprep.o:      module_hinterp_setup.o module_domain_info.o \
                               module_gridded_data.o
```

```
proc_output_variable.o: module_hinterp_setup.o module_gridded_data.o
```

```
proc_store_global_metadata.o: module_hinterp_setup.o
```

```
debug:
```

```
$(RM) *.o *.exe *.mod *.M ; $(MAKE) $(EXE) \  
"MODULELIB      =      $(MODULELIBDEBUG)"      \  
"FFLAGS        =      $(DBFLAGS)" )
```

clean:

```
$(RM) $(FOBJS) $(EXE) *.o *.mod *.M core
```

install: \$(EXE)

```
$(INSTALL) $(EXE) $(INSTALLROOT)/bin/$(EXE)
```

修正が完了したら、コンパイルを実行する。

```
$ ./install_wrfsti.pl --machine=pcintel --install_ui=y
```

次に地理データをインストールする。「SI Geographical Data」を取得する。wget を用いたダウンロードスクリプトがあるのでこれを使用する。ファイアーウォールで wget が成功しない場合は、--passive-ftp オプションをスクリプト中の wget に追加する。

```
$ cd ~/WRF  
$ sh /tmp/wget_geog.shx  
$ mv geog GEOG  
$ cd wrfsti/extdata  
$ rm -rf GEOG  
$ ln -s ../../GEOG
```

これで./wrf_tools で WRFSTI を実行できるが、Vine Linux 3.2 ではスタックサイズが 8192KB に制限されており、この制限にひっかかって Localization に失敗してしまう。ログを見ると、Signal 11 を吐いて強制終了していることが分かる。

```
### wrf_tools  
ERROR: 65280 Command /usr/bin/perl (...)/localize_domain.pl  
  
### localize_domain.log  
ERROR: (...)/gridgen_model.exe ran with signal 11
```

このエラーを回避するためには、スタックサイズを無制限にしてツールを実行する必要がある。次のコマンドを用いる。

```
$ ulimit -s unlimited  
$ ./wrf_tools
```

WRF Version2.1.2

```
$ cd ~/WRF  
$ tar zxvf /tmp/WRFV2.1.2.TAR.gz  
$ cd WRFV2
```

arch/configure.defaults で、5566 行から始まる「Intel xeon i686 ia32 Xeon Linux, ifort compiler (single-threaded, no nesting)」の設定を修正。その後、./configure スクリプトで環境設定を実行し、上記で修正した番号(7)を選択。コンパイルを実行する。

```
### arch/configure.defaults line:5566  
CC = icc  
CPP = ifort -E -free
```

```
$ ./configure  
$ ./compile em_real
```

run ディレクトリに ndown.exe、real.exe、wrf.exe が作成されればコンパイル成功。