GTOOLの使用方法

2015年8月18日

国立環境研究所　山下陽介

このドキュメントは、2005年5月6日に東大気候システム研究センターで行った計算機セミナーのドキュメントを基に作成しました。当時の気候センターの計算機システムはSolarisが主流であったため、Solarisユーザを対象とした内容でしたが、近年の計算サーバではLinuxが主流となってきたため、Linuxユーザを対象に書き換えを行いました。現在の気候モデルの出力では、η座標系 (hybrid σ-p座標系) が主流となったため、鉛直座標系の変換方法について更新しました。

1　GTOOLとは

GTOOLとは、CCSR/NIES AGCMで出力される形式 (GTOOL3形式) のデータを解析、図化するためのツールのことです。コマンドラインツールのみが提供されており、使用するためにはUNIX環境 (Linux、Solaris、Mac等) が必要です。GTOOLでは、データの切出し、データ間の差、空間・時間平均を取るといった簡単な操作をコマンド1つで行うことができ、解析の高速化が可能です。また、鉛直座標系の変換 ($\eta \rightarrow$ p、$\sigma \rightarrow$ p、z $\rightarrow$ p変換) や、パワースペクトル、移動平均、水平発散、水平渦度、流線関数の計算といったデータ解析でよく使われるフィルタコマンドがあらかじめ用意されているため、ある程度のことはプログラムを作成しないでも行うことができます。さらに、GTOOLには描画コマンドがあり、GrADSのようにコントロールファイルを必要としないため、GCMの出力結果をすばやく表示することが可能です。ただ、描画コマンドの出力はGrADSやGMTなどの可視化ソフトに比べてかなり劣るため、発表用の図の作成には向きません。また、GTOOLを使用する人はかなり限定されるため、Webで検索しても参考になるページがほとんどないのも欠点かもしれません。

1.1　GTOOLで扱えるファイル形式

GTOOLではGTOOL3形式 (CCSR/NIES AGCMの出力などはこの形式) という固有のファイル形式のみを取り扱います[*1]。GTOOL3形式はデータ本体の情報等を記述したヘッダとデータ本体で構成され、1時刻に対してヘッダ部＋データ部があり、複数の時刻のデータが順に1つのファイル内に書き込まれた状態になっています。ヘッダ部は16文字 ×64欄の文字データ、データ部は書式なしバイナリデータで、読み込み形式は順アクセスバイナリファイルです[*2]。

ヘッダには、データ日時、格子点数、格子情報ファイル名、データ部のサイズ、欠損値、データ形式、データの名前、作成者、作成日時などが記述されています。ヘッダの詳細はGTOOL、GCM

[*1] GTOOLのうち、データ形式の変換コマンドでは、直接アクセスバイナリファイル、ASCIIファイルを扱えるコマンドもあります。

[*2] Fortranのopen文でaccess='sequential', form='unformatted'として読み込む形式。ヘッダとデータの間には改行コードが入ります。

のマニュアルをご参照下さい。

GTOOL のコマンドでは、ヘッダの情報を利用して処理を行います。格子点の位置情報はヘッダ内には記述されておらず、別に格子情報ファイルを利用します[*3]。したがって、GTOOL のコマンドが正常に動作するためには、ヘッダの情報と格子情報ファイルの内容の両方が正確である必要があります。

1.2 GTOOL コマンドの形式

GTOOL のコマンドは基本的に次のような形式をしています。

```
$ gt____ [入力ファイル名] [オプション]
```

(原則として入力ファイル名とオプションの順番はどうでもよく、オプション名は大文字小文字を区別しません。)

☆オプションの種類

数オプション　：z=1 のように、オプション名=値とする
文字オプション：title:Rain のように、オプション名:値とする
論理オプション：print=t clabel=f のようにオプション名=t/f とする。
(print=t は-print, clabel=f は#clabel のように略記可)

2 GTOOL を使用する準備

2.1 システム環境変数等の設定

GTOOL を使用する前に、以下の変数を設定します。gtool-3.5 は dcl-5.3.1 を利用するので、DCL と GTOOL の両方について設定方法を記述しています。既に GTOOL のインストールが済んでいるシステムでは、以下の設定があらかじめ行われていることもありますので、変更前にシステム管理者等にご確認下さい。

- 環境変数 DCLDIR、GTOOLDIR
  DCLDIR には dcl-5.3.1 の置き場所を、GTOOLDIR には gtool-3.5 の置き場所を指定します。
- 環境変数 GTAXDIR
  格子情報ファイルの置き場所を指定します。デフォルトでは${GTOOLDIR}/lib/gt3 ですので、自分のホームディレクトリなど他の場所に置きたい場合には変更します。この変数を

[*3] GTOOL では、格子点の位置情報をあらかじめ格子情報ファイルとして保存しています。GTOOL はヘッダの中に記述された格子情報ファイル名を基に、保存されたファイルを随時読み出して使用します。実際のファイル名は GTAXLOC.____ですが、ヘッダに記述する格子情報ファイル名は、最後の____部分のみです。

指定するとデフォルトディレクトリは読み込まれなくなるので注意して下さい。

- 環境変数 GTTMPDIR
  GTTMPDIR は、GTOOL が使用する作業領域の場所です。GTOOL の入出力オプションにファイル名を指定しないと、GTOOL はカレントディレクトリの gtool.in や gtool.out というファイルを読み書きします。GTTMPDIR を指定すると、カレントディレクトリの代わりに、GTTMPDIR に gtool.in や gtool.out が作成されるようになります。
- 環境変数 PATH
  PATH には実行ファイルへのパスが書かれているので、その中に GTOOL のバイナリを置くディレクトリを追加します。
- 環境変数 MANPATH
  コマンド man で表示するマニュアルの置き場所を指定します。
- 環境変数 LD_LIBRARY_PATH
  実行時に必要なライブラリの置き場所を指定します[*4]。

設定は次のように行います。DCL は${HOME}/local/dcl-5.3.1 に、GTOOL は${HOME}/local/gtool-3.5 以下に置くものとしていますので、適宜変更して下さい。

- bash を使用する場合、~/.bashrc に次のような記述を書き加えます

```
export DCLDIR=${HOME}/local/dcl-5.3.1
export GTOOLDIR=${HOME}/local/gtool3.5
export GTAXDIR=${GTOOLDIR}/lib/gt3
export PATH=${DCLDIR}/bin:${GTOOLDIR}/bin:${PATH}
export MANPATH=${GTOOLDIR}/man:${MANPATH}
export LD_LIBRARY_PATH=${DCLDIR}/lib:${GTOOLDIR}/lib:${LD_LIBRARY_PATH}
#export GTTMPDIR=${HOME}/work  # 必要な場合のみ
```

- csh、tcsh を使用している場合は、~/.cshrc に次のような記述を書き加えます

```
setenv DCLDIR ${HOME}/local/dcl-5.3.1
setenv GTOOLDIR ${HOME}/local/gtool3.5
setenv GTAXDIR ${GTOOLDIR}/lib/gt3
setenv PATH ${DCLDIR}/bin:${GTOOLDIR}/bin:"${PATH}"
setenv MANPATH ${GTOOLDIR}/man:"${MANPATH}"
setenv LD_LIBRARY_PATH ${DCLDIR}/lib:${GTOOLDIR}/lib:"${LD_LIBRARY_PATH}"
#setenv GTTMPDIR ${HOME}/work  # 必要な場合のみ
```

[*4] ほとんどのプログラムは単独では動作せず、実行時にライブラリを参照するようにコンパイルされています。この実行ライブラリの置き場所が、LD_LIBRARY_PATH です。システムやコンパイラのライブラリなど、複数のプログラムで使用されることが多いライブラリをプログラム本体とは別に管理し実行時にのみ参照することで、個々のプログラムが使用するメモリサイズやファイルサイズなどを小さくできる利点があります。

2.2 GTOOL のインストール

スーパーコンピュータなどで計算されたモデルの出力を取得し、手元の Linux マシンで解析を行うことを想定して、Linux マシンに dcl-5.3.1、gtool-3.5 のインストールを行う方法を記述しています。ここでは、ifort の場合を記述します。既に ifort がインストール済みで、2.1 節の環境変数の設定も終えたものとしています。既にインストールが済んでいるマシンをお使いの場合は、この節を読み飛ばしても構いません。

1. dcl-5.3.1 のインストール
   gtool-3.5 はコンパイル時に dcl-5.3.1 を利用するので、gtool のインストール前に dcl-5.3.1 をインストールしておく必要があります。

```
% cd $DCLDIR
% FC=ifort FFLAGS='-assume byterecl -convert big_endian \
 -O3 -ip -save -zero' \
./configure --prefix=$DCLDIR --disable-gtk --disable-gtk2
% make
% make install
```

2. gtool のマニュアルページの準備
   gtool をコンパイルする前に gtool の man ページをインストールしておきます (man ページは必須ではありませんが)。

```
% cd $GTOOLDIR
% tar xzvf gtooldoc-0.1.tar.gz
% cd gtooldoc-0.1/ja_JP.eucJP
% ./makeman.sh
% cd $GTOOLDIR
% cp -pr gtooldoc-0.1/ja_JP.eucJP/man1 man
```

3. コンパイラ等の設定
   ${GTOOLDIR}/Mkinclude を編集し、システム名 (SYSTEM)、DCL ライブラリの名前 (DCLLIBNAME) を次のように変更します。

```
--------------------------------------------------------------------------
SYSTEM = Linux-ifc14.0
DCLLIBNAME = f77dcl531
--------------------------------------------------------------------------
```

対応する${GTOOLDIR}/libsrc/sysmake/Makedef.Linux-ifc14.0 というファイルを作成します。INTELCOMP は、ifort のインストールされたディレクトリを指定します[*5]。

記述例：

```
-----------------------------------------------------------------------
INTELCOMP       = /opt/intel/composer_xe_2013_sp1.2.144/
INTELTYPE       = intel64
SYSFFLAGS       = -O3 -ip -convert big_endian -assume byterecl \
                  -save -zero
SYSCFLAGS       = -O
SYSLDFLAGS      = -O3 -L$(INTELCOMP)/compiler/lib/$(INTELTYPE)
SYSCPPFLAGS     = -DSYS_Linux -DSYS_UNIX -DCODE_ASCII -DCODE_IEEE
SYSAUTODBL      = -r8
SYSDEBUG        = -u -inline_debug_info
SYSCHECK        = -C -d4
LINKOPT         =

CC              = $(INTELCOMP)/bin/$(INTELTYPE)/icc
FC              = $(INTELCOMP)/bin/$(INTELTYPE)/ifort
LD              = $(INTELCOMP)/bin/$(INTELTYPE)/ifort
AR              = $(INTELCOMP)/bin/$(INTELTYPE)/xiar vru
RM              = rm -f
CP              = cp
MV              = mv -f
LN              = ln -s
RANLIB          = $(INTELCOMP)/bin/$(INTELTYPE)/xiar s
AROBJ           = $(INTELCOMP)/bin/$(INTELTYPE)/xild -r -o
RANOBJ          = touch
CAT             = cat
INSTALL         = cp
MD              = mkdirhier
JLATEX          = bigjlatex
DVI2JPS         = dvi2ps
PRINT           = ltype
PRINTSTAMP      = .print
INDEX           = etags -wx
```

[*5] % which ifort で確認できます。

```
TAGS            = etags
TOUCH           = touch
CPP             = cpp
FPP             =

SYSXLIBDIR      = /usr/lib64
SYSXLIBNAME     = X11
SYSXLIBS        = -L$(SYSXLIBDIR) -l$(SYSXLIBNAME)

PACKFILE        = Linux.ftr
PACKDIR         = $(SRCDIR)/Linux

world:  all

.SUFFIXES : .pac .F

$(PACKFILE):

.F.pac:
        echo "*/ADD NAME="$*.F >> $(PACKFILE)
        cat $< >> $(PACKFILE)
-------------------------------------------------------------------------
```

4. gtool-3.5 のインストール
   次のコマンドで、${GTOOLDIR}/bin/以下に gtool のプログラムがインストールされます。

```
% cd $GTOOLDIR
% make
% make install
```

   古いバージョンの場合には、${GTOOLDIR}/bin/Linux-ifc14.0 以下にインストールされるので、最後に、PATH を通したディレクトリに手動でバイナリを移動します[*6]。

```
% cd $GTOOLDIR/bin/Linux-ifc14.0
% mv * ..
% cd $GTOOLDIR/bin/Linux-ifc14.0
% mv libgtool3-dcl5.a ..
```

[*6] PATH の設定を${GTOOLDIR}/bin/Linux-ifc14.0、LD_LIBRARY_PATH の設定を${GTOOLDIR}/lib/Linux-ifc14.0 に変えても同じです。

3 GTOOL の主要コマンド

3.1 描画コマンド

3.1.1 コンター図の作成

gtcont はコンター図を作成するためのプログラムです。x、y、z オプションでデータから切り取る断面を選択します。

```
% gtcont <file> x=15 (東西方向の 15 個目のグリッドの南北鉛直図)
% gtcont <file> x=15 y=1,10 (y の範囲を 1〜10 個目のグリッドに限定)
% gtcont <file> x=0 (0 を指定すれば、東西平均の緯度ー鉛直プロット)
% gtcont <file> x=0 -exch (-exch オプションで軸を交換)
```

時刻データの限定

```
% gtcont <file> str=2 end=8 (2〜8 番目までをプロット)
% gtcont <file> str=2 end=8 step=2 (2 個ごとにプロット)
```

コンター間隔を変える

```
% gtcont <file> cont=5 (コンター間隔を 5 にする)
% gtcont <file> cont=5,10 (10 ごとに太線にする)
% gtcont <file> cont=5,10 range=0,100 (0〜100 の間のみ描く)
```

値による塗り分け

```
% gtcont <file> tone=0,100 pat=14 (0〜100 までトーンをかける)
% gtcont <file> tone=-100,100,200 pat=14,24 (2 つのパターンで塗り分け)
% gtcont <file> color=20 (20 色で塗り分ける)
```

★東西平均した図で西風領域のみにトーンをかけるには?

その他

```
% gtcont <file> color=20 map=1 (海岸線を重ねてみる)
% gtcont <file> color=20 polar=t sp=t map=1 (ポーラーマップ)
% gtcont <file> title:'Averaged temperature' (タイトルを付ける)
% gtcont <file> dset:'MIROC3.2' (右上の Dataset 名を指定)
% gtcont <file> -print ps:out.ps (out.ps に出力)
```

3.1.2 ベクトル図の作成

gtvect は 2 つのデータからベクトル図を作成するコマンドです。

```
% gtvect <file1> <file2> (1 つ目が x 軸、2 つ目が y 軸成分)
% gtvect <file1> <file2> intv=3,3 (3 点おきにプロット)
% gtvect <file1> <file2> fact=0.5 (ベクトルの長さを 0.5 倍に)
% gtvect <file1> <file2> range=-10,10 (最大値 10 としてスケーリング)
```

☆ベクトルとコンターを同時に描くコマンドもある

```
% gtvecon u v Z intv=3,3 x=30,100 y=1,37 map=1
(東西方向の 30〜100 番目と南北方向の 1〜37 番目のグリッドを選択)
```

3.2 表示コマンド

3.2.1 キャラクタ表示

gtshow は、データをキャラクタ形式で標準出力に表示するコマンドです。

```
% gtshow <file> z=1
(z=1 の面のみを切り出して出力)
% gtshow <file> tstr=3 tend=7 tstep=2
(3、5、7 番目のレコードのみを出力)
```

☆ヘッダの情報だけを表示するオプションは便利

```
% gtshow -item <file>
(変数名、時刻、軸情報ファイル名、軸のサイズを簡略表示)
% gtshow -s <file>
(最初の 1 レコードだけを表示することもできる)
```

3.3 フィルタコマンド

3.3.1 データの切り出し、時間平均

gtsel はデータの切出し、断面の作成などを行うコマンド、gtavr は時間平均の作成を行うコマンドです。

```
% gtsel <file1> x=0 ; gtcont (東西平均)
% gtsel <file1> x=30,100 y=1,37 out:<file2>
(グリッドを切り出す)
% gtsel <file1> z=5,10 str=1 end=32 step=4 ; gtsel z=0 ; gtcont (これは?)
```

```
% gtavr <file1> out:<file2> str=2 end=10 step=2 (2〜10 まで 2 つおき平均)
% gtavr <file1> out:<file2> ostep=30 (30 個ごとの平均をとる)
```

★最初の 30 日平均と次の 30 日平均の東西風の図を作り、西風領域だけをカラーで表示したい。

3.3.2 簡単な計算コマンド

gtadd(和)、gtsub(差)、gtmlt(積)、gtdiv(商)、gtlog(対数)、gtsqrt(平方根) コマンドなどがあります。２つのデータに別々のファクタとオフセットを作用させることも可能です。

```
% gtsub <file1> <file2> out:<file3> fact1=0.1 ofs1=1 ofs2=3
    ( (0.1*<file1> + 1 ) - (1*<file2> + 3) )
```

★最初の 30 日と次の 30 日の東西風の差を取ったら?

```
% gtlog <file1> out:<file2> (logA の計算)
```

★ gtedy(擾乱を計算するコマンド) の出力を使って擾乱の起こりやすい場所を調べたい。

```
% gtedy -x Z out:z-edy (東西平均からの擾乱成分を計算)
% gtmlt z-edy z-edy ; gtavr str=1 end=30 ; gtsqrt ; gtcont map=1(これは?)
```

☆データに定数を掛けるコマンドもある

```
% gtset Ps_hPa out:Ps_Pa fact=100 unit:'Pa'
```

(地表気圧の単位を hPa から Pa に変換する)

3.3.3 移動平均、パワースペクトル

移動平均を取るための gtmavr、パワースペクトルを計算するための gtpwspct、gtstspct といったデータ解析でよく使われるフィルタコマンドがあります。

```
% gtmavr <file1> out:<file2> -x peri=10 (x 軸の方向に 10 点移動平均)
% gtpwspct <file1> out:<file2> -x (x 軸方向のパワースペクトル)
% gtstspct <file1> out:<file2> (時空間パワースペクトル)
```

(第 1 次元を経度、第 3 次元を時間とする必要あり)

3.3.4 発散、渦度、流線関数

球面調和関数展開を用いた水平発散 (gthdiv)、水平渦度 (gthvor) の計算、南北風からの子午面流線関数の作成 (gtstrm) といったことも行えます。

% gthdiv u v out:<file> (水平発散の計算)

% gthvor u v out:<file> (水平渦度の計算)

% gtstrm v ps:Ps str=1 end=1 ; gtcont

(子午面流線関数の計算。file1 は南北風、file2 は地表面気圧データ)

3.3.5 温位、相当温位、仮温度、飽和比湿、相対湿度

% gttheta <file1> ps:<file2> out:<file3> (温度から温位を計算)

% gttheta <file1> ps:<file2> out:<file3> -inv (温位から温度を計算)

% gtthetae t:<file1> q:<file2> ps:<file3> out:<file4> (相当温位を計算)

(t で温度、q で比湿、ps で地表面気圧データを指定する)

% gttv t:<file1> q:<file2> out:<file3> (温度、比湿から仮温度を計算)

% gtqsat t:<file1> ps:<file2> out:<file3> (飽和比湿)

% gtrelh q:<file1> t:<file2> ps:<file3> out:<file4> (相対湿度)

3.3.6 座標変換

鉛直座標系の変換

z 座標から p 座標への変換 (gtz2p)、σ 座標から p 座標 (gts2p) への変換が可能です。

% gtz2p <file1> p:<file2> out:<file3> -apnd

(3 次元の気圧データを p で指定。追加書き込みを行う)

% gts2p <file1> ps:<file2> out:<file3> #misout

(地表面気圧が指定気圧より低い場合に外挿する)

☆ gts2p コマンドでは、η 座標から p 座標への変換を行うこともできる。

% gts2p <file1> ps:<file2> out:<file3> plev:GPLV31

(GPLV31 の気圧面に変換する)

% gts2p <file1> ps:<file2> out:<file3> plev:GPLV31 logl=t

(log レベルで補間)

水平座標系の変換

入力したファイルを、x、y オプションで指定した軸情報ファイルの座標系に変換します。

% gtintrp <file1> x:GLON128 y:GGLA64 out:<file2>

(T42 の解像度の水平グリッドに変換する)

☆解像度の細かいデータから解像度の粗いデータに変換する場合、gtintrp よりも平均操作を同時に行う gtintrp2 の方が滑らかな場ができる。

3.4 データ形式変換コマンド

3.4.1 GTOOL3 形式から GrADS 形式に変換

gtwdir は GTOOL3 形式から GrADS 形式[*7]に変換するコマンドです。[*8]

```
% gtwdir <file1> out:<file2> -r4
(実数 4 バイトで出力。GrADS 形式にする場合には、このオプションを使用)
% gtwdir <file1> out:<file2> -i2
(整数 2 バイトで出力することもできる)
```

3.4.2 GrADS 形式から GTOOL 形式に変換

gtrdir は GrADS 形式のファイルを GTOOL3 形式のファイルに変換するコマンドです。他の形式の気象データを一旦 GrADS 形式にしてから GTOOL で読めるようにしたい場合などに使用します。x、y、z オプションで各軸の格子情報ファイル名称を指定します。

```
% gtrdir <file1> out:<file2> -r4 x:GLON144 y:GLAT73 z:NCEPL17
(NCEP/NCAR 再解析データの場合)
% gtrdir <file1> out:<file2> -r4 x:GLON144 y:GLAT73 z:SFC1
(地表面データを変換)
% gtrdir <file1> out:<file2> -r4 x:144 y:73 z:17
(格子情報ファイルがない場合、とりあえず格子数を指定)
```

☆ヘッダ情報を編集することもできる

```
% gtset T out:T-out title:'Temperature' dset:'NCEP/NCAR' \
  item:'T' unit:'K' utim:'DAY' styp=2
(変数名を T、単位を K、styp=2 で鉛直方向を log スケールに)
```

4 GTOOL のマニュアルなど

- 本体付属のドキュメント (${GTOOLDIR}/doc/Guide.tex)
- UNIX の man コマンドで表示できるようにしたもの (gtooldoc-0.1.tar.gz)
- GTOOL のマニュアルを HTML で書き直したページ
  https://www.riam.kyushu-u.ac.jp/taikai/lab/others/Gtool/Guide.html

[*7] GrADS 形式は、1 レコードが 4 バイトの単精度浮動小数点形式で記述された直接アクセスバイナリファイルです。

[*8] 出力されるデータの endian は入力データと同じです。big/little 変換をするには-r4r, -i2r のようなオプションをつけます。

付録 A　GTOOL のインストール方法 (Solaris 8–9)

ここには、2005 年に東大気候システム研究センターで計算機セミナーを行った際の内容を掲載しています。実際のインストール場所などは、システムによって異なりますので、ご了承下さい。

- DCL のインストール
  GTOOL をインストールするには、先に DCL をインストールする必要があります。

  DCL5.2 以前の場合
  ${DCLDIR}/sys/Mkinclude. システム名 を${DCLDIR}にコピーして
  ```
  % make
  % make install
  ```

  DCL5.3 の場合
  ```
  % ./configure --prefix=インストール先
  % make
  % make install
  ```

- 環境変数の設定
  GTOOL のインストール先は環境変数 GTOOLDIR で決まるので、GTOOLDIR を設定します。また、GTOOL のインストールの際に DCL ライブラリを使用できるように環境変数 LD_LIBRARY_PATH の設定を行います。通常は LD_LIBRARY_PATH に既にシステムが使用するライブラリのパスが入っているので、この値に追加します。

  ```
  % setenv DCLDIR /usr/local/gtool
  % setenv LD_LIBRARY_PATH $DCLDIR/lib:"$LD_LIBRARY_PATH"
  ```

- Mkinclude の編集
  ${GTOOLDIR}/Mkinclude の最初の方にシステムを設定する部分 (SYSTEM=の部分) があるので、自分のシステムに合わせて編集します。Solaris サーバでは SYSTEN=Solaris です。この指定により、${GTOOLDIR}/libsrc/sysmake/Makedef.${SYSTEM} が読み込まれるようになります。Makedef.Solaris の SYSXLIBNAME で指定されたディレクトリは climate サーバでは存在しないので/usr/openwin/lib に変更します。コンパイラを変更したい場合には FC、CC、LD を変更し、コンパイラオプションを変更したい場合には SYSFFLAGS、SYSCFLAGS などを変更します。
- コンパイルとインストール

  ```
  % make dirs (インストール先のディレクトリを作成)
  % make
  % make install (${GTOOLDIR}以下にインストールされる)
  ```

A.1 GTOOL を使用するための設定 (Solaris 8–9)

Solaris の場合、GTOOL を使用する前に、次の変数を設定します。

- 実行ファイルへのパス
  ${HOME}/.cshrc のシェル変数 PATH には実行ファイルへのパスが書かれているので、その中に GTOOL のバイナリがあるディレクトリを追加します。実行ファイルへのパスは、climate7 では/usr/local/gtool/bin、他では/usr/local/gt3-dcl5/bin/Solaris なので、次のように記述します。

  記述例

```
if ("$HOST" == "climate7") then
  set path = ( $path /usr/local/dcl-5.0.2/bin \
               /usr/local/gtool/bin )
else
  set path = ( $path /usr/local/dcl-5.0.2/bin \
               /usr/local/gt3-dcl5/bin/Solaris )
endif
```

- 環境変数 GTOOLDIR
  GTOOLDIR には GTOOL がインストールされたディレクトリを指定します。climate7 では/usr/local/gtool、他では/usr/local/gt3-dcl5 なので、次のように記述します。

  記述例

```
if ("$HOST" == "climate7") then
  setenv  GTOOLDIR  /usr/local/gtool
else
  setenv  GTOOLDIR  /usr/local/gt3-dcl5
endif
```

- 環境変数 GTAXDIR
- 環境変数 GTTMPDIR

付録 B　GTOOL のインストール方法 (Solaris 11)

ここには、最近の Solaris 11 で GTOOL をインストールする手順について記述しています。まず、gcc-4.8.2(+gfortran) をインストールし、それから dcl-5.3.1 と gtool-3.5 をコンパイル/インストールすることを想定しています。

B.1　gcc/gfortran のインストール

pkg コマンドを使い、gcc-4.8.2 をインストールします。バージョンはインストール時点のものです。

```
# pkg install -v gcc-48
# /usr/bin/gcc --version (gcc のバージョンを確認)
# /usr/bin/gfortran --version (gfortran のバージョンを確認)
```

B.2　環境変数の設定とインストール

コンパイルを行う前に、GTOOLDIR、DCLDIR、PATH 等を設定します。2.1 節をご参照下さい。

- DCL5.3.1 のインストール

  ```
  % cd dcl-5.3.1
  % make clean.all
  ```
  csh/tcsh の場合
  ```
  % setenv FC gfortran
  % setenv FFLAGS '-O -fconvert=big-endian'
  % ./configure --prefix=インストール先 --disable-gtk --disable-gtk2
  ```
  bash の場合
  ```
  % FC=gforran FFLAGS='-O -fconvert=big-endian' \
    ./configure --prefix=/usr/local/dcl-5.3.1  \
    --disable-gtk --disable-gtk2
  ```
  csh/tcsh/bash 共通
  ```
  % make
  % make install
  ```

- コンパイラ等の設定

  ${GTOOLDIR}/Mkinclude を編集し、システム名 (SYSTEM)、DCL ライブラリの名前 (DCLLIBNAME) を次のように変更します。

```
-------------------------------------------------------------------------
SYSTEM = Sol11
DCLLIBNAME = f77dcl531
-------------------------------------------------------------------------
```

対応する${GTOOLDIR}/libsrc/sysmake/Makedef.Sol11 というファイルを作成します。

記述例：

```
-------------------------------------------------------------------------
SYSFFLAGS       = -O -fconvert=big-endian
SYSCFLAGS       =
SYSLDFLAGS      = -lm
SYSCPPFLAGS     = -DSYS_Sun -DSYS_UNIX -DCODE_ASCII -DCODE_IEEE \
                  -DE_EXTERNAL
SYSAUTODBL      = -r8
SYSDEBUG        = -g
SYSCHECK        = -u -C
LINKOPT         =

CC              = gcc
FC              = gfortran
LD              = gfortran
AR              = ar vru
RM              = rm -f
CP              = cp
MV              = mv -f
LN              = ln -s
RANLIB          = touch
AROBJ           = ld -r -o
RANOBJ          = touch
CAT             = cat
INSTALL         = cp
MD              = mkdirhier
JLATEX          = bigjlatex
DVI2JPS         = dvi2ps
PRINT           = ltype
PRINTSTAMP      = .print
INDEX           = etags -wx
```

```
TAGS            = etags
TOUCH           = touch
CPP             = cpp
FPP             =

SYSXLIBDIR      = /usr/X11/lib
SYSXLIBNAME     = X11
SYSXLIBS        = -L$(SYSXLIBDIR) -l$(SYSXLIBNAME) -lsocket -lnsl -lm

PACKFILE        = Sun.ftr
PACKDIR         = $(SRCDIR)/Sun

world:  all

.SUFFIXES : .pac .F

$(PACKFILE):

.F.pac:
        echo "*/ADD NAME="$*.F >> $(PACKFILE)
        cat $< >> $(PACKFILE)
-------------------------------------------------------------------------
```

- gtool-3.5 のインストール
  次のコマンドで、${GTOOLDIR}/bin/以下に gtool のプログラムがインストールされます。

```
% cd $GTOOLDIR
% make
% make install
```

付録 C　GTOOL3 形式から GrADS 形式への変換プログラムの例

```
program main
  integer(4), parameter :: imax = 128, jmax = 64, kmax = 31
  character(len=16)     :: head(64)
  real(4)               :: GD(imax,jmax,kmax)
  character(len=100)    :: fname1, fname2
  integer(4)            :: iflag

  read(5,*) fname1, fname2
  open(10, file=fname1, form='unformatted', status='old', &
&   access='sequential')

  open(51, file=fname2, form='unformatted', status='unknown', &
&   access='direct', recl=imax*jmax*kmax*4)

  iflag = 1
  !c+++ GTOOL3 形式データ読み込み
1 read(10, end=2) head
  read(10, err=999) GD

  !c+++ GrADS 形式データ書き出し
  write(51, rec=iflag) GD
  iflag = iflag + 1
  goto 1
2 continue

  stop
  999 write(6,*) 'Error: DO NOT READ DATA ...'
end program main
```

付録 D　GTOOL3 形式で書き出しを行うプログラムの例

```
subroutine wgt(jfile, imax, jmax, kmax, vmiss, d, hitem, htitl, hunit, &
& hdset, dfmt, time, tdur, htunit, jdate, haxisx, haxisy, haxisz, ios)
  !c+++ [input]
  integer(4), intent(in) :: jfile  !! 出力ファイル番号
  integer(4), intent(in) :: imax, jmax, kmax !! x, y, z 軸のサイズ
  real(8), intent(in) :: vmiss !! 欠損値
  real(8), intent(in) :: d(imax,jmax,kmax)
  character(len=*), intent(in) :: hitem !! 変数名
  character(len=*), intent(in) :: htitl !! タイトル
  character(len=*), intent(in) :: hunit !! 単位
  character(len=*), intent(in) :: hdset !! データセット名
  character(len=*), intent(in) :: dfmt  !! データフォーマット
  real(8), intent(in) :: time  !! 時刻
  real(8), intent(in) :: tdur  !! 代表時間
  character(len=*), intent(in) :: htunit !! 時刻単位
  integer(4), intent(in) :: jdate (6) !! 日付
  character(len=*), intent(in) :: haxisx !! x 軸名称
  character(len=*), intent(in) :: haxisy !! y 軸名称
  character(len=*), intent(in) :: haxisz !! z 軸名称
  !c+++ [output]
  integer(4), intent(out) :: ios !! end code
  !c+++ [internal work]
  integer(4) :: i
  character(len=16) :: head(64)
  integer(2) :: jdates(7) !! input from getdat, gettim
  integer(4) :: jdaten(6)
  character(len=16*2) :: htitlz
  character(len=16)   :: huser
  real(4), allocatable :: d4(:,:,:)
  real(8), allocatable :: d8(:,:,:)
  logical :: opend

  inquire(unit=jfile, opened=opend)
  if (.not. opend) then
    write(6, '(a, i3)') 'Error: not opend, unit = ', jfile
```

```
  ios = 1
  return
endif

call getdat(jdates(1), jdates(2), jdates(3))
call gettim(jdates(4), jdates(5), jdates(6), jdates(7))
jdaten(1:6) = jdates(1:6)
call getenv('USER', huser)

head(1:64) = ' '
write(head(1), '(i16)') 9010
head(2) = hdset
head(3) = hitem
head(12) = '1   '
head(13) = '1   '
htitlz   = htitl
head(14) = htitlz(1:16)
head(15) = htitlz(17:32)
head(16) = hunit
write(head(25), '(i16)') int(time)
write(head(28), '(i16)') int(tdur)
head(26) = htunit
write(head(27), '(i4.4, 2i2.2, 1x, 3i2.2)') jdate(1:6)
head(29) = haxisx
write(head(30), '(i16)') 1
write(head(31), '(i16)') imax
head(32) = haxisy
write(head(33), '(i16)') 1
write(head(34), '(i16)') jmax
head(35) = haxisz
write(head(36), '(i16)') 1
write(head(37), '(i16)') kmax
head(38) = dfmt
do i = 39, 43
  write(head(i), '(e16.7)') vmiss
enddo
write(head(44), '(i16)') 1
```

```
  write(head(46), '(i16)') 0
  write(head(47), '(e16.7)') 0.
  write(head(48), '(i16)') 0
  write(head(60), '(i4.4, 2i2.2, 1x, 3i2.2)') jdaten(1:6)
  head(61) = huser
  write(head(62), '(i4.4, 2i2.2, 1x, 3i2.2)') jdaten(1:6)
  head(63) = huser
  write(head(64), '(i16)') imax*jmax*kmax

  !c++ ヘッダ書き出し
  write(jfile, iostat=ios) head
  if (ios /= 0) then
    write(6, '(a, i2)') 'Error: write gtool3 header, unit = ', jfile
    return
  endif
  !c++ データ書き出し
  if (dfmt == 'UR4') then !! 単精度浮動小数点
    allocate(d4(imax,jmax,kmax))
    d4(1:imax,1:jmax,1:kmax) = d(1:imax,1:jmax,1:kmax)
    write(jfile, iostat=ios) d4(1:imax,1:jmax,1:kmax)
    deallocate(d4)
  else if (dfmt == 'UR8') then !! 倍精度浮動小数点
    allocate(d8(imax,jmax,kmax))
    d8(1:imax,1:jmax,1:kmax) = d(1:imax,1:jmax,1:kmax)
    write(jfile, iostat=ios) d8(1:imax,1:jmax,1:kmax)
    deallocate(d8)
  else
    write(6, *) 'Error:', dfmt, 'is not supported yet.'
    stop 2
  endif

  if (ios /= 0) then
    write(6, '(a, i2)') 'Error: write gtool3 data, unit = ', jfile
  endif

  return
end subroutine wgt
```