10.PW_WRF_ARWPOST_2

WRF/ARWpostのデータ（プレーンバイナリ形式）から可降水量を計算する

複数の時刻について処理（入力，計算，出力）をする

復習

復習：データの所在

```
$ ls
/work00/DATA/WRF.RW3A/HD01/RW3A.ARWpost.DAT/basic_p/ARWpost_RW3A.00.03.05.05.000
0.01/
```

復習：入力データのCTLファイル

```
$ ls
/work00/DATA/WRF.RW3A/HD01/RW3A.ARWpost.DAT/basic_p/ARWpost_RW3A.00.03.05.05.000
0.01/*ctl

/work00/DATA/WRF.RW3A/HD01/RW3A.ARWpost.DAT/basic_p/ARWpost_RW3A.00.03.05.05.000
0.01/RW3A.00.03.05.05.0000.01.d01.basic_p.01HR.ctl
```

CTLファイルの内容

```
$ cat /work00/DATA/WRF.RW3A/HD01/RW3A.ARWpost.DAT/basic_p/ARWpost_RW3A.0
0.03.05.05.0000.01/*ctl
dset ^RW3A.00.03.05.05.0000.01.d01.basic_p.01HR_%y4-%m2-%d2_%h2:%n2.dat
options  byteswapped template
undef 1.e30
title  OUTPUT FROM WRF V4.1.5 MODEL
pdef  599 599 lcc  27.000  130.500  300.000  300.000  32.00000  27.00000
130.50000   3000.000   3000.000
xdef 1469 linear  120.56867   0.01351351
ydef 1231 linear   18.56023   0.01351351
zdef   30 levels
1000.00000
 990.00000
 980.00000
 970.00000
 960.00000
 950.00000
 940.00000
 930.00000
 920.00000
 910.00000
 900.00000
 880.00000
 860.00000
 840.00000
 820.00000
 800.00000
 750.00000
 700.00000
 650.00000
 600.00000
 550.00000
```

```
 500.00000
 450.00000
 400.00000
 350.00000
 300.00000
 250.00000
 200.00000
 150.00000
 100.00000
tdef   73 linear 00Z12AUG2021      60MN
VARS   30
U             30  0  x-wind component (m s-1)
V             30  0  y-wind component (m s-1)
W             30  0  z-wind component (m s-1)
Q2             1  0  QV at 2 M (kg kg-1)
T2             1  0  TEMP at 2 M (K)
U10            1  0  U at 10 M (m s-1)
V10            1  0  V at 10 M (m s-1)
QVAPOR        30  0  Water vapor mixing ratio (kg kg-1)
QCLOUD        30  0  Cloud water mixing ratio (kg kg-1)
QRAIN         30  0  Rain water mixing ratio (kg kg-1)
HGT            1  0  Terrain Height (m)
RAINC          1  0  ACCUMULATED TOTAL CUMULUS PRECIPITATION (mm)
RAINRC         1  0  RAIN RATE CONV (mm per output interval)
RAINNC         1  0  ACCUMULATED TOTAL GRID SCALE PRECIPITATION (mm)
RAINRNC        1  0  RAIN RATE NON-CONV (mm per output interval)
XLAND          1  0  LAND MASK (1 FOR LAND, 2 FOR WATER) (-)
PBLH           1  0  PBL HEIGHT (m)
HFX            1  0  UPWARD HEAT FLUX AT THE SURFACE (W m-2)
QFX            1  0  UPWARD MOISTURE FLUX AT THE SURFACE (kg m-2 s-1)
LH             1  0  LATENT HEAT FLUX AT THE SURFACE (W m-2)
SST            1  0  SEA SURFACE TEMPERATURE (K)
ept           30  0  Equivalent Potential Temperature (K)
sept          30  0  Saturated Equivalent Potential Temperature (K)
pressure      30  0  Model pressure (hPa)
height        30  0  Model height (km)
tk            30  0  Temperature (K)
theta         30  0  Potential Temperature (K)
rh            30  0  Relative Humidity (%)
slp            1  0  Sea Levelp Pressure (hPa)
dbz           30  0  Reflectivity (-)
ENDVARS
```

復習：CTLファイルから読み取ったレコード番号

最初の時刻について，各変数のレコード番号を下記に記載する。

```
01-30 U        30  0  x-wind component (m s-1)
31-60          30  0  y-wind component (m s-1)
61-90 W        30  0  z-wind component (m s-1)
91 Q2           1  0  QV at 2 M (kg kg-1)
92 T2           1  0  TEMP at 2 M (K)
93 U10          1  0  U at 10 M (m s-1)
94 V10          1  0  V at 10 M (m s-1)
95-124 QVAPOR  30  0  Water vapor mixing ratio (kg kg-1)
125-154 QCLOUD 30  0  Cloud water mixing ratio (kg kg-1)
.....
```

左端の数値が, 3列目の鉛直層数を積算することで求めた，各変数が収録されているレコード番号である。

復習：最初の時刻だけ読む

必要な変数（QVAPOR）だけ読む（最初の時刻）

```
$ vi READ_WRF_2.1.F90
```

```
PROGRAM READ_WRF
CHARACTER(LEN=1000):: INDIR, INFLE
CHARACTER(LEN=2000):: IN
REAL,DIMENSION(:,:,:),ALLOCATABLE::QV

INDIR="/work00/DATA/WRF.RW3A/HD01/RW3A.ARWpost.DAT/basic_p/ARWpost_RW3A.00.03.05
.05.0000.01/"
INFLE="RW3A.00.03.05.05.0000.01.d01.basic_p.01HR_2021-08-15_00:00.dat"


IM=599; JM=599; KM=30
ALLOCATE(QV(IM,JM,KM))

IN=TRIM(INDIR)//TRIM(INFLE)

PRINT *,"INPUT: ",TRIM(IN)

OPEN(11,FILE=IN,ACTION="READ",form="unformatted",access="direct",recl=IM*JM*4)

IREC=94
DO K=1,KM
IREC=IREC+1
READ(11,rec=IREC)QV(:,:,K)
END DO !K

WRITE(6,*)
WRITE(6,*)'IM/2=',IM/2, 'JM/2=',JM/2
WRITE(6,*)'QV(IM/2,JM/2,1)=',QV(IM/2,JM/2,1)

CLOSE(11)
END PROGRAM READ_WRF
```

```
$ ift
```

```
$ ifort -convert big_endian -traceback -CB -assume byterecl READ_WRF_02.F90 -o
READ_WRF_02.EXE
```

```
$ READ_WRF_2.1.EXE
 INPUT:
 /work00/DATA/WRF.RW3A/HD01/RW3A.ARWpost.DAT/basic_p/ARWpost_RW3A.00.03.05.05.00
 00.01/RW3A.00.03.05.05.0000.01.d01.basic_p.01HR_2021-08-15_00:00.dat

 IM/2=         299 JM/2=         299
 QV(IM/2,JM/2,1)=  1.8941429E-02
```

復習：WRFデータを用いた可降水量の計算

FORTRANプログラム

PW_WRF.F90

```
PROGRAM PW_WRF
CHARACTER(LEN=1000):: INDIR, INFLE, ODIR, OFLE
CHARACTER(LEN=2000):: IN, OUT
INTEGER,PARAMETER::KM=30
REAL P(KM)
REAL,ALLOCATABLE::QV(:,:,:),PW(:,:)
REAL,PARAMETER::UNDEF=1.E30

DATA P/1000.00000,990.00000,980.00000,970.00000,960.00000,&
950.00000,940.00000,930.00000,920.00000,910.00000,900.00000,&
880.00000,860.00000,840.00000,820.00000,800.00000, 750.00000,&
700.00000,650.00000,600.00000,550.00000,500.00000,450.00000,&
400.00000,350.00000,300.00000,250.00000,200.00000,150.00000,&
100.00000/

INDIR="/work00/DATA/WRF.RW3A/HD01/RW3A.ARWpost.DAT/basic_p/ARWpost_RW3A.00.03.05
.05.0000.01/"
INFLE="RW3A.00.03.05.05.0000.01.d01.basic_p.01HR_2021-08-15_00:00.dat"

ODIR="./"
OFLE="PW_2021-08-15_00:00.dat"

IM=599; JM=599
ALLOCATE(QV(IM,JM,KM), PW(IM,JM))



IN=TRIM(INDIR)//TRIM(INFLE)
PRINT *,"INPUT: ",TRIM(IN)
OPEN(11,FILE=IN,ACTION="READ",form="unformatted",access="direct",&
recl=IM*JM*4)
IREC=94
```

```
DO K=1,KM
IREC=IREC+1
READ(11,rec=IREC)QV(:,:,K)
END DO !K
CLOSE(11)



do k=1,km
p(k)=p(k)*100.0  !hPa -> Pa
end do

PW=0.0
do k=1,km-1
PW(:,:)=PW(:,:)+(QV(:,:,k)+QV(:,:,k+1))*(p(k)-p(k+1))/2.0
end do

WHERE(PW<UNDEF)
PW=PW/9.8
ELSEWHERE
PW=UNDEF
ENDWHERE

PRINT *,PW(IM/2,JM/2)

OUT=TRIM(ODIR)//TRIM(OFLE)
PRINT *,"OUTPUT: ",TRIM(OUT)
OPEN(21,FILE=OUT,form="unformatted",access="direct",recl=IM*JM*4)
IREC=1
WRITE(21,rec=IREC)PW(:,:)
CLOSE(21)

END PROGRAM PW_WRF
```

GrADSで作図して確認

CTLファイル：PW.CTL

```
dset ^PW_%y4-%m2-%d2_%h2:%n2.dat
options  byteswapped template
undef 1.e30
title  OUTPUT FROM WRF V4.1.5 MODEL
pdef  599 599 lcc  27.000  130.500  300.000  300.000  32.00000  27.00000
 130.50000   3000.000   3000.000
xdef 1469 linear  120.56867   0.01351351
ydef 1231 linear   18.56023   0.01351351
zdef   1 levels  10000.0
tdef   1 linear 00Z15AUG2021      60MN
VARS  1
PW        1  0  PRECIPITABLE WATER (mm)
ENDVARS
```

描画スクリプト：CHECK.PW.GS

```
'open PW.CTL'
'set t 1'
'q dims'
date=sublin(result,5)
say date

'cc'
'set gxout shaded'
'd PW'
'cbarn'

FIG='PW_CHECK.PDF'
'gxprint 'FIG
say 'OUTPUT: 'FIG
```

GrADSで作図

```
$ grads -bcp
```

```
ga-> CHECK.PW.GS
Notice: Implied interpolation for file PW.CTL
 Interpolation will be performed on any data displayed from this file
T is fixed     Time = 00Z15AUG2021  T = 1
OUTPUT: PW_CHECK.PDF
ga-> quit
```

演習

今回作成するプログラムの仕様

仕様1

- 2番目の時刻のデータを読み込んで可降水量を計算し，プレーンバイナリファイルで計算結果を出力する
- CTLファイルを書き換える
- 計算結果を図示する

仕様2

- 入力データに記録されている全ての時刻に関して可降水量を計算し，プレーンバイナリファイルで計算結果を出力する
- 時刻ごとに出力ファイルを分ける
- CTLファイルを書き換える
- 計算結果を図示する

仕様1のヒント

PW_WRF.F90の変更点

入力データは各時刻ごとに個別のファイルに記録されているので，FORTRANプログラムにおいて次の時刻のファイル名を指定すればよい。

```
INFLE="RW3A.00.03.05.05.0000.01.d01.basic_p.01HR_2021-08-15_00:00.dat"

OFLE="PW_2021-08-15_00:00.dat"
```

PW.CTLの変更点

開始時刻00Z15AUG2021で，時刻に関するデータ数1となっているので，データ数を変更する（2番目の列）

```
tdef   1 linear 00Z15AUG2021      60MN
```

仕様2のヒント

日付処理はFortranでもできるが，linuxのdateというコマンドの方が，簡潔に処理できるので，今回はこれを活用する。ポイントは下記の通り。

- 日付処理に関するところはdateコマンドにやらせて，その結果をFortranのプログラムに渡す
- 結果を渡すには，linuxのシェルスクリプトとFortranのnamelistを使うのが便利

dateコマンドの例

実際にコマンドを打ち込んで，実行結果を逐一確認すること

```
$ date
2024年  1月 27日 土曜日 12:11:14 JST
```

```
$ date -R
Sat, 27 Jan 2024 12:11:16 +0900
```

```
$ date +"%Y-%m-%d_%H:%M:%S"
2024-01-27_12:13:41
```

```
$ T1=$(date +"%Y-%m-%d_%H:%M:%S")
$ echo $T1
2024-01-27_12:15:30
```

```
$ yyyy1=2021; mm1=08; dd1=21
$ start=${yyyy1}/${mm1}/${dd1}
$ jsstart=$(date -d${start} +%s) # 1970年1月1日0時UTCからのうるう秒を考慮しない秒数
$ echo $jsstart SECONDS
1629471600 SECONDS
$ jhstart=$(expr $jsstart / 3600) #時間単位
$ echo $jhstart HOURS
452631 HOURS
```

```
$ yyyy1=2021; mm1=08; dd1=21; hh1=0
$ start="${yyyy1}/${mm1}/${dd1} ${hh1}:00:00"
$ jsstart=$(date -d"${start}" +%s) # 1970年1月1日0時UTCからのうるう秒を考慮しない
秒数
$ echo $jsstart SECONDS
1629471600 SECONDS
$ jhstart=$(expr $jsstart / 3600) #時間単位
$ echo $jhstart HOURS
452631 HOURS
```

```
$ yyyy2=2021; mm2=08; dd2=21; hh2=1
$ end="${yyyy2}/${mm2}/${dd2} ${hh2}:00:00"
$ jsend=$(date -d"${end}" +%s) # 1970年1月1日0時UTCからのうるう秒を考慮しない
秒数
$ jhend=$(expr $jsend / 3600) #時間単位
$ echo $jhend HOURS
452632 HOURS
```

詳しい使用法については下記参照

https://hydrocul.github.io/wiki/commands/date.html

https://gitlab.com/infoaofd/lab/-/blob/master/LINUX/01.BASH/LINUX_DATE.md?ref_type=heads

bashスクリプトにおけるループと加算の例

実際にスクリプトを打ち込んで，実行結果を逐一確認すること

1.LOOP_SAMPLE.sh

```
IS=1; IE=5
I=$IS

while [ $I -le $IE ]; do
echo "I= $I"
I=$(expr $I + 1)
done #I
```

```
$ chmod u+x 1.LOOP_SAMPLE.sh
```

```
$ 1.LOOP_SAMPLE.sh
I= 1
I= 2
I= 3
I= 4
I= 5
```

dateコマンドによる日付処理の例

実際にスクリプトを打ち込んで，実行結果を逐一確認すること

2.DATE_SAMPLE.sh

```
yyyy1=2021; mm1=08; dd1=12; hh1=0
IS=1; IE=73

start="${yyyy1}/${mm1}/${dd1} ${hh1}:00:00"
jsstart=$(date -d"${start}" +%s)

jhstart=$(expr $jsstart / 3600)

I=$IS
while [ $I -le $IE ]; do

date_out=$(date -d"${start} ${I}hour" +"%Y%m%d%H")

yyyy=${date_out:0:4}; mm=${date_out:4:2}; dd=${date_out:6:2}
hh=${date_out:8:2}

echo "$I   $date_out   $yyyy $mm $dd $hh"

I=$(expr $I + 1)
done
```

```
$ chmod u+x 2.DATE_SAMPLE.sh
```

```
$ 2.DATE_SAMPLE.sh
1   2021081201   2021 08 12 01
.....
73   2021081501   2021 08 15 01
```

Fortranのnamelistの使用例

実際にスクリプトとプログラムを打ち込んで，実行結果を逐一確認すること

BASHスクリプト

3.NAMELIST_SAMPLE.sh

```
echo

# NAMELISTファイルの作成
NML=3.NAMELIST.TXT
echo MMMMM CREATE $NML
cat <<EOF > $NML
&para
ODIR="OUT_PW"
OFLE="PW_2021-08-12_00.BIN"
&end
EOF
echo

# NAMELISTファイルの内容確認
echo MMMMM CHECK $NML
cat $NML
echo

# FORTRANプログラムの存在確認
F90=3.NAMELIST_SAMPLE.F90
echo MMMMM CHECK IF WE HAVE $F90
if [   -f $F90 ];then echo F90 SOURCE FILE,$F90    ;fi
if [ ! -f $F90 ];then echo NO SUCH FILE,$F90;exit 1;fi
echo

# プログラムの実行ファイル名の設定
EXE=$(basename $F90 .F90).EXE

# 過去に作成された実行ファイルが存在していたら念のため削除しておく
echo MMMMM DELETE $EXE IF EXISTS
rm -vf $EXE
echo

# FORTRANプログラムのコンパイル（実行ファイルの作成）
echo MMMMM COMPILE $F90
ifort -traceback -CB -assume byterecl $F90 -o $EXE
if [   -f $EXE ];then echo EXECUTABLE FILE,$EXE       ;fi
if [ ! -f $EXE ];then echo NO SUCH FILE,   $EXE;exit 1;fi
echo

# プログラムの実行
echo MMMMM EXECUTE $EXE
$EXE < $NML
echo

exit 0
```

FORTRANプログラム

3.NAMELIST_SAMPLE.F90

```
CHARACTER ODIR*100, OFLE*100

NAMELIST /para/ODIR,OFLE

READ(5,NML=para)

PRINT '(A,A)',"ODIR: ",TRIM(ODIR)
PRINT '(A,A)',"OFLE: ",TRIM(OFLE)

STOP
END
```

INTEL FORTRAN コンパイラの設定

```
$ ift

:: initializing oneAPI environment ...
.....
:: oneAPI environment initialized ::
```

3.NAMELIST_SAMPLE.sh の実行

```
$  3.NAMELIST_SAMPLE.sh

MMMMM CREATE 3.NAMELIST.TXT

MMMMM CHECK 3.NAMELIST.TXT
&para
ODIR="OUT_PW"
OFLE="PW_2021-08-12_00.BIN"
&end

MMMMM CHECK IF WE HAVE 3.NAMELIST_SAMPLE.F90
F90 SOURCE FILE,3.NAMELIST_SAMPLE.F90

MMMMM DELETE 3.NAMELIST_SAMPLE.EXE IF EXISTS
`3.NAMELIST_SAMPLE.EXE' を削除しました

MMMMM COMPILE 3.NAMELIST_SAMPLE.F90
EXECUTABLE FILE,3.NAMELIST_SAMPLE.EXE

MMMMM EXECUTE 3.NAMELIST_SAMPLE.EXE
ODIR: OUT_PW
OFLE: PW_2021-08-12_00.BIN
```

NAMELISTファイルの内容確認

```
$ cat 3.NAMELIST.TXT
&para
ODIR="OUT_PW"
OFLE="PW_2021-08-12_00.BIN"
&end
```

日付処理とFORTRANプログラムの組み合わせ

BASHスクリプト

4.DATE_LOOP.sh

```


echo

F90=4.DATE_LOOP.F90
echo MMMMM CHECK IF WE HAVE $F90
if [   -f $F90 ];then echo F90 SOURCE FILE,$F90    ;fi
if [ ! -f $F90 ];then echo NO SUCH FILE,$F90;exit 1;fi
echo

EXE=$(basename $F90 .F90).EXE

echo MMMMM DELETE $EXE IF EXISTS
rm -vf $EXE
echo

echo MMMMM COMPILE $F90
ifort -traceback -CB -assume byterecl $F90 -o $EXE
if [   -f $EXE ];then echo EXECUTABLE FILE,$EXE       ;fi
if [ ! -f $EXE ];then echo NO SUCH FILE,   $EXE;exit 1;fi
echo

echo MMMMM SET INPUT DIRECTORY
INDIR="/work00/DATA/WRF.RW3A/HD01/RW3A.ARWpost.DAT/basic_p/ARWpost_RW3A.00.03.05
.05.0000.01/"
if [ ! -d $INDIR ]; then echo NO SUCH DIR.,$INDIR;exit 1;fi

echo MMMMM CREATE OUTPUT DIRECTORY, $ODIR
ODIR="OUT_PW/"
mkdir -vp $ODIR

echo MMMMM SET NAMELIST FILE NAME
NML=4.DATE_LOOP_NML.TXT
echo NAMELIST FILE:  $NML

echo MMMMM SET RECORD NUMBER OF QVAPOR
IREC=94
```

```
yyyy1=2021; mm1=08; dd1=12; hh1=0
IS=1; IE=72

start="${yyyy1}/${mm1}/${dd1} ${hh1}:00:00"

jsstart=$(date -d"${start}" +%s)
jhstart=$(expr $jsstart / 3600)

I=$IS
while [ $I -le $IE ]; do

date_out=$(date -d"${start} ${I}hour" +"%Y%m%d%H")

yyyy=${date_out:0:4}; mm=${date_out:4:2}; dd=${date_out:6:2}
hh=${date_out:8:2}

echo MMMMMMMMMM $I   $yyyy $mm $dd $hh

echo MMMMM SET INPUT AND OUTPUT FILE NAMES
INFLE="RW3A.00.03.05.05.0000.01.d01.basic_p.01HR_${yyyy}-${mm}-${dd}_${hh}:00.da
t"
OFLE="PW_${yyyy}-${mm}-${dd}_${hh}:00.00.dat"

echo MMMMM CHECK IF INPUT FILE REALLY EXISTS
IN=$INDIR$INFLE
if [ ! -f $IN ];then echo NO SUCH FILE,$IN;exit 1;fi

echo MMMMM CREATE $NML
cat <<EOF > $NML
&para
INDIR="${INDIR}"
INFLE="${INFLE}"
ODIR="${ODIR}"
OFLE="${OFLE}"
IREC=$IREC
&end
EOF
echo

echo MMMMM EXECUTE $EXE
$EXE < $NML
echo

I=$(expr $I + 1)
echo

done #I

exit 0
```

FORTRANプログラム

4.DATE_LOOP.F90

```
CHARACTER(LEN=500)::INDIR, INFLE, ODIR, OFLE
```

```
INTEGER IREC

NAMELIST /para/INDIR,INFLE,ODIR,OFLE,IREC

READ(5,NML=para)

PRINT '(A,A)',"INDIR: ",TRIM(INDIR)
PRINT '(A,A)',"INFLE: ",TRIM(INFLE)
PRINT '(A,A)',"ODIR: ",TRIM(ODIR)
PRINT '(A,A)',"OFLE: ",TRIM(OFLE)
PRINT '(A,I5)',"IREC= ",IREC

STOP
END
```

実行例

```
$ 4.DATE_LOOP.sh

MMMMM CHECK IF WE HAVE 4.DATE_LOOP.F90
F90 SOURCE FILE,4.DATE_LOOP.F90

.....

MMMMM EXECUTE 4.DATE_LOOP.EXE
INDIR:
/work00/DATA/WRF.RW3A/HD01/RW3A.ARWpost.DAT/basic_p/ARWpost_RW3A.00.03.05.05.000
0.01/
INFLE: RW3A.00.03.05.05.0000.01.d01.basic_p.01HR_2021-08-15_00:00.dat
ODIR: OUT_PW/
OFLE: PW_2021-08-15_00:00.00.dat
IREC=    94
```

付録

上達のためのポイント

エラーが出た時の対応の仕方でプログラミングの上達の速度が大幅に変わる。

ポイントは次の3つである。

1. エラーメッセージをよく読む
2. エラーメッセージを検索し，ヒットしたサイトをよく読む
3. 変数に関する情報を書き出して確認する

エラーメッセージは，プログラムが不正終了した直接の原因とその考えられる理由が書いてあるので，よく読むことが必要不可欠である。

記述が簡潔なため，内容が十分に理解できないことも多いが，その場合**エラーメッセージをブラウザで検索**してヒットした記事をいくつか読んでみる。

エラーの原因だけでなく，**考えうる解決策**が記載されていることも良くある。

エラーを引き起こしていると思われる箇所の**変数の情報**や**変数の値そのものを書き出して，期待した通りにプログラムが動作しているか確認する**ことも重要である。

エラーの場所が特定できれば，エラーの修正の大部分は完了したと考えてもよいほどである。

エラーメッセージや検索してヒットするウェブサイトは英語で記載されていることも多いが，**重要な情報は英語で記載されていることが多い**ので，よく読むようにする。

重要そうに思われるが，一回で理解できないものは，PDFなどに書き出して後で繰り返し読んでみる。どうしても**内容が頭に入らないものは印刷してから読む**。

参考

CTLファイルの概要

別途資料 (BINARY_DATA.pptx)も参照のこと

https://gitlab.com/infoaofd/lab/-/blob/master/FORTRAN/PROGRAM_2022/BINARY_DATA_GRADS.pdf

- `^`: カレント・ディレクトリ，自分が現在いるディレクトリ）を意味する記号 この場合したがって，ctlファイルとデータファイルが同じ場所に保存されていることを仮定している (^を実際にデータがあるディレクトリに書き換えることで変更可能)
- `template`: ファイル名の指定にひな型 (template)を使う
- `byteswapped`: バイナリデータはビッグエンディアンである。

  https://gitlab.com/infoaofd/lab/-/blob/master/FORTRAN/PROGRAM_2022/BINARY_DATA_GRADS.pdf
- `undef 1.e30`: 値が存在しない場所には，ダミーの値として10の30乗が入っている。
- `pdef`: (GrADSでのみ使用される)不等間隔の格子データを，緯度・経度上のデータに変換するための情報
- `xdef`: 東西方向のデータ並びに関する情報※
- `ydef`: 南北方向のデータ並びに関する情報※
- `tdef`: 時間方向のデータ並びに関する情報
- `zdef`: 鉛直方向のデータ数
- `VARS`: 保存されている変数の数

※ここでのxdef, ydefに記載されているデータ数はGrADSで描画するときのみ使用される値で，**実際にはpdefに記載されている数のデータがファイルに保存されている**。

templateの書式

- %y4 : 年 (4桁の整数)
- %m2: 月 (2桁の整数)
- %d2: 日 (2桁の整数)
- %h2: 時 (2桁の整数)
- %n2: 分 (2桁の整数)

例えば%y4-%m2-%d2_%h2:%n2だと，データファイルの名前の中で

2021-08-15_00:00

のような形式で，日時がしていされていることを意味する。

```
RW3A.00.03.05.05.0000.01.d01.basic_p.01HR_%y4-%m2-%d2_%h2:%n2.dat
```

であれば，RW3A.00.03.05.05.0000.01.d01.basic_p.01HR2021-08-15_00:00.datのようなデータファイルが存在しているはずである。

今回の例では，下記を実行して確かめることができる。

```
$ ls /work03/2021/sakagami/WRF.RW3A.00.03.05.05/ARWpost_RW3A
.00.03.05.05.0000.01
```

pdefの書式

```
pdef XSIZE YSIZE LCCR(LCC) YLAT XLON X Y SLAT1 SLAT2 SLON DIS_X DIS_Y
```

`XSIZE`, `YSIZE`: X, Y方向のデータ数

(詳しくは本資料末尾参照)

xdef (ydef)の書式

```
xdef データ数 データの並べ方 西の端のデータの経度　東西方向の格子間隔(単位は度)
```

- linear＝データの並びは等間隔である

zdefの書式

```
zdef データ数 データの並べ方 下端のデータの座標　上端のデータの座標 （この場合の単位は気圧）
```

- levels = データの間隔は不等間隔なので，このすぐ下に座標一覧を示す

tdefの書式

```
tdef データ数　データの並べ方 最初のデータの時刻 時間間隔
```

変数一覧の書式

`U`: 変数の名前

```
U             30  0  x-wind component (m s-1)
```

- 鉛直方向に30個データ有
- 変数の名前はx-wind componentで，単位は(m s-1)である。

要点

- ビッグエンディアンで記録されている
- 東西方向に599個，南北方向に599個のデータがある※
- 鉛直方向には30個のデータがある。
- 欠損値は1e30（10の30乗）としている

※**pdefに記載されいている数値が実際のデータ数**で，xdef, ydefの値はGrADSで描画するときのみ使用される。