netCDF データの規約 (属性) についての調査

平成 11 年 10 月 23 日 (第 1 版)

赤堀浩司
akahori@cse.nagoya-u.ac.jp

Contents

1 はじめに

1.1 情報源

調査にあたっては、netCDF のホームページ http://www.unidata.ucar.edu/packages/netcdf/ 以下の情報を利用した。

基本となるものは、netCDF を提供する Unidata が定めた次の規約 (ドキュメントと呼ぶ) である。

- NetCDF-3 User's Guide for Fortran, Russ 他著, 1997, http://www.unidata.ucar.edu/packages/netcdf/guidef/, の 8. Attribute Convention の冒頭および 8.1 Attribute Conventions. (NetCDF-3 User's Guide for C も調査したが、属性に関しては両者はほぼ同じである).

  このドキュメントの ver.2.4 の石岡圭一氏による翻訳「netCDF ユーザーズガイド (version 2.4), (FORTRAN 対応部分の日本語訳)」http://www.gfd-dennou.org/arch/netcdf/netcdf-jp/SIGEN.htm. もバージョンの違いに注意して参照・転載した.

NetCDF では、利用者グループが適当な手続きに従って新たな規約を定めることができる。その情報は、上述のホームページの「NetCDF conventions」の項目以下から得ることができる。

- COARDS Convention, (Cooperative Ocean/Atmosphere Research Data Service が提供, 1995, http://ferret.wrc.noaa.gov/noaa_coop/coop_cdf_profile.html)
- GDT Convention, (Gregory, Drach, Tett による規約, 1999 年 3 月の Ver.1.3 が最新, http://www-pcmdi.llnl.gov/drach/GDT_convention.html)
- ICSPAT Conventions
- NCAR-RAF Conventions
- NUWG Conventions
- PMEL-EPIC Conventions
- Proposals for coordinate conventions and coordinate conventions postings (ongoing discussion)

2 情報源の概略

2.1 ドキュメント

以下の「属性 (attribute)」について説明している。

```
units
long_name
valid_min
valid_max
valid_range
scale_factor
add_offset
_FillValue
missing_value
signedness
C_format
FORTRAN_format
title
history
Conventions
```

2.2 COARDS の規約の項目

以下の項目について説明している。必ずしも「属性」ではないことに注意。次節の GDT はこの規約を意識している。

```
File Name:
Coordinate Variables:
Global attributes:
Data Variable attributes:
long_name
scale_factor
add_offset
_FillValue
missing_value
Units attribute:
Other attribute:
Variable names:
Rectilinear coordinate systems, only:
Number of dimensions:
Coordinate variable names:
Order of dimensions:
Data type:
Coordinate value ordering:
Coordinate Variable Attributes:
Time or date dimension:
Climatological time:
Vertical (height or depth) dimension:
Latitude dimension:
Longitude diminsion:
```

2.3 GDT の規約の項目

以下の項目について説明している。「属性」ではないことに注意。属性の説明は、各項目の中で適宜行なわれており、また、Appendix A に一覧表がある。文章中では COARDS データと相反する規約について注意が書いてある。量は膨大。

```
1  Purposes
2  Filename
3  Data types
4  Attributes
5  Global attributes
6  Variable names
7  Data variables
8  Coordinate variables
9  Axes and dimensionality of a data variable
10  Coordinate systems
11  Units
12  Physical quantity of a variable
13  Topology of an axis
14  Longitude dimension
15  Latitude dimension
16  Vertical (height or depth) dimension
17  Component variables
18  Associated variables
19  Bundles
20  Boundary variables
21  Representation of subgrid variation
```

```
22  Contracted dimensions
23  Time variables and intervals
24  Relative time
25  Absolute time
26  Gregorian calendar
27  Non-Gregorian calendars
28  Multiple time axes and climatological time
29  Invalid values in a data variable
30  Missing values in a data variable
31  Compression by gathering
32  Compression using a scale and offset
A  Attributes
B  Methods of representing subgrid variation
C  Modifications to udunits.dat
D  Long names for quantities
```

2.4 ICSPAT 規約

Imperial College Space and Atmospheric Physics Group (ICSPAT) の規約。この規約は 1994 年 の Q 時系列解析で採用されたとあるが、規約の具体的な内容は載っていない。

2.5 NCAR-RAF 規約

National Center for Atmospheric Research の Research Aviation Facility の規約。1994 年。飛行機の時系列データに特化した規約を定めている。サンプルデータあり。

```
Dimensions:
Time:
Global attributes:
Variable attributes:
```

2.6 NUWG 規約

1992 年に作られた NetCDF Users Working Group (NUWG) の規約。サンプルデータあり。ミーティングをしていろいろなことを議論しているが、Web ページで見る限りでは 1996 年以降は活動を停止している。現在のところ、この規約については調査していない。

```
General Conventions approved by NUWG（少ない）
Conventions for Surface Observations（少ない）
Conventions for Upper Air Observations（少ない）
Conventions for Georeferenced Gridded Data（多い）
Conventions for Satellite Data（ちょっと多い）
```

2.7 PMEL-EPIC 規約

NOAA の Pacific Marine Environmental Laboratory (PMEL) が、海洋データに対して EPIC ソフトウエアパッケージを使うために作った規約。サンプルプログラム・データあり。未調査。

```
General Conventions
Axes
   Longitude Axis
   Latitude Axis
```

```
        Depth Axis
        Time Axis
      Variables
      Attributes

      Oceanographic Data Types:
  Conventions for Profile Type Data , e.g., CTD, XBT, Bottle data
  Conventions for Time Series Type Data , e.g., Buoy data
  Conventions for Track-line data, e.g., Shipboard ADCP Data (preliminary)
```

2.8 Proposals for coordinate conventions and coordinate conventions postings (ongoing discussion)

メイルの一覧がある。未調査。

3 ドキュメントに記載された属性

ここでは、ドキュメントの 8.1 節を掲載し、そこに COARDS および GDT が定める規約の該当部分との比較調査の結果を示す。

アンダースコア (‘_’) から始まる名前は netCDF ライブラリの使用のために予約されている. netCDF ファイルを加工する一般の多くのアプリケーションは, 標準的な属性の慣習を仮定しており, そうしないだけの正当な理由が無い限りは, これらのとりめに従うことを強く勧める. 以下に, 推奨される標準的な属性 (便利であることがすでに示されている) の名前および意味のリストを示す. これらのうち, 数値データが仮定され, 文字データとしては使われないものもある (例えば, `units`, `valid_range`, `scale_factor`) ことに注意が必要である.

- `units`

  ドキュメント 変数データに対して使われている単位を指定する文字列. Unidata は, 自由に取得可能なルーチンのライブラリを提供しており, それによって文字列と単位記述のバイナリ形式との間の変換を行うことができ, さらにそのバイナリ形式においては, いろいろと便利な操作を実行できる. このライブラリはいくつかの netCDF アプリケーションにおいて使われている. 推奨される単位構文を使えば, 整合単位で表現されたデータを, 算術演算できる一般的な単位に自動的に変換できる. より多くの情報については, Appendix A [単位] を参照.

  COARDS 基本的にドキュメントに同じく udunits に従うことを推奨する. その複数形も使える. ただし, その規定から ”degrees” を削除し、”evel”,”layer”,”sigma_level” を追加する。経度を表すときは, ”degrees_east”,”degree_east”,”degrees_E”, ”degree_E” を使うことを推奨する. 経度は ”degrees_north”, ”degree_north”,”degrees_N”,”degree_N” を使うことを推奨する.

  GDT 基本的にドキュメントに同じく udunits に従うことを推奨する. ドキュメントとの違いは、Appendix C で示しており、今のバージョンでは、degrees は認めない. year と month は認めない. calendar_month と calender_year は, お互いあるいは他の時間単位との変換ができない. ただし, カレンダー年と, 12 カレンダー月の倍数との間では変換可能.

  (11) この規約のユーザは、ユーザ自身の単位を定義すべきではない。これは、ユーザのファイルを配布しにくくするからである。Unidata に新しい単位の要求を要望すべきである。

  (12) 変数の物理量に対する単位説明。属性 `units` においては大・小文字の区別は重要。この属性は、量が無次元 (純粋な数) でないならば必須である。

  (14) 経度を表わす座標変数は属性 `units` を含まなければならない。デフォルトはない。

  (15) 緯度を表わす座標変数は属性 `units` を含まなければならない。デフォルトはない。

(16) COARDS と異なり、鉛直座標は属性 units を含む必要はない。そのため、正の方向の判断が COARDS と異なることに注意。

また、時間に関係した次の項目で単位に関する記述がある: (23) 時間変数と間隔について、(24) 相対時間について、(25) 絶対時間について、(26) グレゴリオ暦について、(27) 非グレゴリオ暦について。

- long_name

ドキュメント 長い記述的な名前. これは, 例えば, プロットのラベルづけに使うことができる. もし, 変数に属性 long_name が割り当てられていなければ, 変数名がデフォルトとして使われる.

COARDS ドキュメントと同様.

GDT この属性はオプション。

- valid_min

ドキュメント この変数に対する有効な値の最小値を示すスカラー.

COARDS 記載なし

GDT 基本的にドキュメントに同じ。ただし、GDT が定める属性に関連して細かい約束事がある (例えば boundary 変数など。)

- valid_max

ドキュメント この変数に対する有効な値の最大値を示すスカラー.

COARDS 記載なし

GDT 上の valid_min の項目を参照のこと

- valid_range

ドキュメント この変数に対する有効な値の最小値および最大値を示す 2 つの数値のベクトルであり, 属性 valid_min および属性 valid_max の両方の値を指定するのに等しい. これらのどれもが有効範囲を定義する. もし, valid_min または valid_max のどちらかが定義されている場合には valid_range を定義してはならない.

一般的なアプリケーションは, 有効範囲を外れた値を欠損として扱う. valid_range, valid_min, および valid_max のそれぞれの型は, その変数の型に合っていなければならない (ただし, byte 型データを除く. この場合は, これらの属性の型は, 意図する範囲を指定するために, 符号つきの integral 型であってもよい).

valid_min, valid_max, および valid_range のどれも定義されていない場合には, 一般的なアプリケーションは, 有効範囲を以下のように定義する. データ型がバイト型で, _FillValue が陽に定義されていない場合は, 有効範囲はすべての可能な値を含む. 一方, 有効範囲は, _FillValue を以下のように除外する. もし, _FillValue が正ならば, それが有効な最大値を定義し, そうでなければ, それは有効な最小値を定義する. 整数型については, _FillValue とこの有効な最大または最小値との間には 1 だけ違いがある. 浮動小数点型については, この違いは丸め誤差を許すために, 表現可能な最小値 (最も下位のビットが 1) の 2 倍の値である.

COARDS 記載なし

GDT 座標変数には有効範囲外の値は許されない。しかし、有効範囲を示す属性を、境界のないセルを示すために境界変数内で用いることができる。

valid_min か valid_max のどちらかを定義したら、valid_range を定義すべきではない。

byte 型の扱いは推奨しないので、この型に対する特別の扱いは定義していない。

- scale_factor

ドキュメント ある変数についてこの属性が与えられている場合, データは, それにアクセスするアプリケーションによって読み込まれる際にこのファクター倍される.

COARDS ドキュメントと同様.

GDT ドキュメントと同様.

- add_offset

ドキュメント ある変数についてこの属性が与えられている場合, データは, それにアクセスするアプリケーションによって読み込まれた後にこの数が加えられる. scale_factor および add_offset の両方の属性が与えられている場合には, データはまずスケールされ, その後にオフセットが加えられる. 属性 scale_factor および属性 add_offset を同時に使うことによって, 簡単なデータ圧縮を行うことができる; 有効桁数の小さい浮動小数点データを小さな整数として netCDF ファイルに格納することによって. スケールされたデータを書く際には, アプリケーションは, まずオフセットを引いて, その後でスケールファクターで割ればよい.

scale_factor および add_offset を圧縮のために使う際には, 関係する変数 (圧縮されたデータを格納している) は, 普通, byte 型または short 型である; 一方, 解凍されたデータは float または double 型になるようにしたい. というわけで, 属性 scale_factor および属性 add_offset は両方とも, 解凍されたデータの持つべき型 (例えば, float または double 型) でなければならない.

COARDS 基本的にドキュメントと同じ. ただし, NOAA cooperative standard は以下のようにより厳しいものとなっている。

属性 scale_factor と属性 add_offset が関連した変数と同じデータタイプの場合には, 特に制限は設けない。パックされていないデータは, パックされたデータと同じデータタイプと仮定される.

属性 scale_factor と属性 add_offset が (パックされたデータを含む) 関連した変数と異なるデータタイプの場合には, ファイル内の関連した変数は単に byte, short, または long のタイプかもしれない。属性 scale_factor と属性 add_offset (これらはデータタイプが一致しないといけない) は float または double でなければならない。この属性のデータタイプは、アンパックされたデータの意図されたタイプに一致すべきである. (long を float にアンパックすることは, 潜在的な精度の消失があるので薦められない.)

GDT 座標変数も圧縮してよい.

- _FillValue

ドキュメント 属性_FillValue は, 変数に割り当てられているディスクスペースを前もって埋めるために使われるフィル値を指定する. このような前もって埋める動作は, NCSFIL を使ってノーフィルモードが設定されていない場合になされる. 詳細については, [書き込みのためのフィルモードの設定: NCSFIL] の節を参照. フィル値は, まだ書かれていない値を読むと返される. _FillValue が定義される場合, それはスカラーで, 変数と同じ型でなければならない. _FillValue は通常, 有効範囲の外側にあり, それゆえ, 一般のアプリケーションではでは欠損として扱われる. しかし, _FillValue を有効範囲内にして, 他の有効な値のように扱えるようにするのは正当なことである. 変数の型に対するデフォルトのフィル値が適当であれば, 独自の_FillValue を定義する必要はない. しかし, byte データ型に対しては, デフォルトのフィル値を使うことは勧められない. この属性の値を変えた場合, その変更された値は引き続く書き込みにのみ適用されることに注意されたい; すなわち, それまでに書かれたデータは変更されない. より多くの情報については, [フィル値] の節を参照.

COARDS データ変数が scale_value 属性によってパックされている場合には_FillValue も同様にパックされる。

GDT データ変数を属性 scale_factor と属性 add_offset を使ってパックする際、属性_FillValue の値はパックされない。アンパックの際には、実行前に _FillValue の存在をチェックすること。この扱いは **COARDS** とは異なる。

- missing_value

ドキュメント この属性は, ライブラリまたは慣習に従った一般的なアプリケーションで特別に扱われるわけではないが, 記録として便利なこともあり, また特定のアプリケーションで使われることもありうる. 属性 missing_value は, 欠損データを示す値を含むスカラーまたはベクトルでありうる. これらの値はすべて, 一般的なアプリケーションが欠損として扱えるように, 有効範囲の外側にあるべきである. より多くの情報については, [フィル値] の節を参照.

COARDS 2 種類の欠損値を区別する必要があるときは、属性_FillValue と属性 missing_value の存在が役立つ。属性 missing_value のデータタイプは、それが属すデータ変数のデータタイプと一致する。データ変数が scale_value 属性によってパックされている場合には missing_value フラグも同様にパックされる。NOAA の規約では、missing_value と _FillValue を区別して特別に解釈することは推奨しない。

GDT データ変数にだけ適用できる。座標変数に使っては駄目。 属性_FillValue とは違い、netCDF API で特別な方法で使われない。データ変数が属性 scale_factor と属性 add_offset を使ってパックされる際は、属性 missing_value もパックされる。この規定は **COARDS** と異なっており、missing_value と _FillValue を区別する特別の解釈を与えている。

- signedness

ドキュメント 価値のなくなった属性である; もともとは, バイト値が符号つきまたは符号なしのどちらで扱われるべきかを指定するように作られたものである. 属性 valid_min および属性 valid_max をこの目的のために使うことができる. 例えば, バイト変数を非負の値として格納したい場合には, valid_min = 0 および valid_max = 255 を使うことができる. この属性は, netCDF ライブラリでは無視される.

COARDS とくになし

GDT とくになし

- C_format

ドキュメント この変数の値をプリントする C アプリケーションにおいて使われるフォーマットを与える文字列である. 例えば, 変数が有効数字 3 桁の精度しかないことが分かっているなら, 属性 C_format を"%.3g"と定義するのが適当である. ncdump ユーティリティプログラムは, この属性を, それが定義されている変数に対して使う. このフォーマットは, スケーリングを行う属性 scale_factor および属性 add_offset のあるなしにかかわらず, スケールされた (内部の) 型および値に対して適用される.

COARDS とくになし

GDT とくになし

- FORTRAN_format

ドキュメント この変数の値をプリントする FORTRAN アプリケーションにおいて使われるフォーマットを与える文字列である. 例えば, 変数が有効数字 3 桁の精度しかないことが分かっているなら, 属性 FORTRAN_format を"(G10.3)"と定義するのが適当である.

COARDS とくになし

GDT この属性は、変数および座標軸に対して用いることができる。

- title

ドキュメント グローバル属性で, そのデータセットの中身を簡単に記述する文字列.

COARDS とくになし

GDT とくになし

- history

ドキュメント 履歴の検査のためのグローバル属性. これはそのファイルを修正したプログラムの各呼び出しに対する行からなる文字配列である. たちのよい一般的な netCDF アプリケーションは, 日付, 時刻, ユーザ名, プログラム名, およびコマンド引数, を含む行を追加する.

COARDS 必須ではないが、netCDF ファイル内に含まれるデータの履歴を記録するために推奨される。

GDT グローバル属性 history は、全てのデータへ適用されることが仮定されている。個々のデータ変数はそれ自体の属性 history を持ち追加情報を提供する。これらの属性は、グローバル属性よりも優先権を持つ。

属性 quantity と属性 subgrid に正規の形式または（　）内におさめられないようなものを属性 history に記録する。なお、データ変数は、属性 institution や属性 production も持っており、これらはデータが最初にどのように得られたかを示す。属性 history, 属性 institution, 属性 production は、ファイル内のデータ変数を区別するために用いるべきではない。

- Conventions

ドキュメント ある変数についてこの属性が与えられている場合, 'Conventions' は, グローバル属性で, そのファイルが従っている慣習の名前を表す文字配列である; それは, 分野に特化した慣習の集合を記述した文書を貯蔵しているディレクトリからの, 相対ディレクトリ名として解釈できる文字列の形式になっている. これによって, 慣習の階層構造が可能となり, 場所を与えて, 慣習の記述および例を, それを定義している機関およびグループに維持してもらうことができる. 慣習のディレクトリ名は, 現在, ホストマシン ftp.unidata.ucar.edu の put/netcdf/Conventions/ディレクトリから相対的に解釈される. 代わりに, 慣習を記述する文章が維持されている WWW サイトを指定するために, 完全な URL 指定子を使ってもよい.

例えば, NUWG という名前のグループにおいて, 次元名, 変数名, 必要な属性, およびある分野に特化したデータ構造のための netCDF 表現, に対する一まとまりの慣習についての合意ができているのなら, その合意されたとりきめを記述する文章を, 慣習のディレクトリのサブディレクトリ NUWG/ 以下のファイルに保管しておくことができる. これらの慣習に従ったファイルは, "NUWG"という値を持つグローバル属性 Conventions を含むことになる.

後になって, もしそのグループが, NUWG データの特定の部分集合, 例えば, 時系列データ, に対する追加の慣習について合意したなら, その追加の慣習の記述はサブディレクトリ NUWG/TIME_series/ に保管されることになる; また, これらの追加の慣習に従ったファイルは, "NUWG/Time_series"という値を持つグロー バル属性 Conventions を使って, このファイルが NUWG の慣習および追加的な NUWG の時系列の慣習, に従っていることを示すことになる.

COARDS とくに目新しい点はない。

GDT ドキュメントと同様。ただし、ファイルを生成したアプリケーションによって用いられた規約の付録のバージョン番号を記録するために、float の属性 appendices を付加することを推奨する。

4 COARDS を吟味

ここでは、COARDS が独自に定める規約に注目した調査結果を示す。

ファイル名 NetCDF ファイルには拡張子 ".nc" をつけるべき.

座標変数 次元名が変数名と一致するような 1 次元 netCDF 変数は「座標変数」と認識される.

グローバル属性 とくに目新しい点はない。

データ変数属性

long_name とくに目新しい点はない。

scale_factor とくに目新しい点はない。

add_offset NOAA cooperative standard は、この属性を次のように厳密に定めている。

属性 scale_factor と属性 add_offset が関連した変数と同じデータタイプの場合には, 特に制限は設けない。パックされていないデータは, パックされたデータと同じデータタイプと仮定される.

属性 scale_factor と属性 add_offset が (パックされたデータを含む) 関連した変数と異なるデータタイプの場合には, ファイル内の関連した変数は単に byte, short, または long のタイプかもしれない。属性 scale_factor と属性 add_offset (これらはデータタイプが一致しないといけない) は float または double でなければならない。この属性のデータタイプは、アンパックされたデータの意図されたタイプに一致すべきである. (long を float にアンパックすることは, 潜在的な精度の消失があるので薦められない.)

_FillValue とくに目新しい点はない。

missing_value 2 種類の欠損値を区別する必要があるときは、(属性_FillValue と属性 missing_value の存在は) 役に立つ。属性 missing_value の netCDF データタイプは、それが記述するデータ変数の netCDF データタイプと一致する。したがって、データ変数が scale_value 属性によってパックされている場合には missing_value フラグも同様にパックされる。これは属性 _FillValue に対しても同様である。NOAA の規約では、missing_value と _FillValue を区別して特別に解釈することは推奨しない。

単位属性 とくに目新しい点はない。

他の属性 ファイルは、この規約にない他の属性をもつことがある。それらはこの規約を破るようなことを表現しない。アプリケーションプログラムは、認識しない属性を無視すること。

変数名 変数名は文字で始まり, 文字, 数字,_ で構成される. 変数名は大・小文字を区別しないことが推奨される. 変数が重複しないように、同じファイル内で大・小文字を区別しない同じ名前を用いるべきでない.

rectlinear な座標系に限る netCDF 変数内の点の空間/時間位置は, 座標軸から関連する値によって構成された単純に並んだ要素で表されるべきである。それで、例えば, 座標位置が他の非座標変数あるいは方程式から推測されねばならない曲線座標系はこの netCDF プロファイルでは標準化されていない。

次元の数 全ての netCDF 変数は 1, 2, 3, 4 次元空間のいづれかで定義されだろう。

ひとつの点が意味をなすところでは、位置は座標変数としてコード化される. 例えば、鉛直プロファイルにおける緯度と経度を、このひとつの点の緯度座標変数および経度座標変数として用いるのは自然なことである。4 次元以上の netCDF ファイルを作る必要がある場合, 追加される次元は CDL において表現されるように空間次元と時間次元の左に加えられることが推奨される. 例えば 5 個目のパラメータ値座標の表現では次の形式が推奨される。

```
float my_variable(param_value, time, hight, lat, lon);
```

座標変数名 座標変数名は他の netCDF 変数と同様にして名付ける.

次元の順序 いくつかの或いは全ての変数の次元が "date or time" (T), "height or depth" (Z),"latitude" (Y), あるいは"longitude" (X) の解釈をもつ場合には、これらの次元は相対的に T, Z, Y, X の順で定義されなければならない。

データタイプ 座標および非座標変数のデータタイプは限定されていない (byte, short, long, float, double の全てが認められている). 機能的に "byte" と同一である "char" は標準のデータタイプでは禁じられていないが, netCDF は新しいバージョンでその動作を変更する予定であるため, 推奨しない.

座標変数の順序 座標変数の座標値は単調増加または単調減少でなければならない。しかし、座標値は等間隔である必要はない。座標変数においては欠損値は許されない。

座標変数の属性 座標変数が longitude, latitude, depth, elevation, date, time の値を含む場合には、単位属性を必ずつけなければならない。座標変数の方向の決定に使うためである. 属性 `long_name` はオプションであるが, 指定した場合 netCDF ファイルの透過性および自己記述性が高まる. 属性 `_FillValue` と属性 `missing_value` を座標変数に用いることはできない。

Time あるいは **date** 次元 時間を表す座標変数は常に陽に属性 `units` を含まなくてはならない. デフォルト値はない. 時間座標変数であることは `units` によってのみ確認される. 属性 `units` は, ユニデータの udunits パッケージバージョン 1.7.1. で推奨されるフォーマットの文字タイプである.

時間に関して許容される `units` の一覧が udunits.dat ファイルにある. もっとも広く用いられている string (およびその短縮形) には, day(d), hour(hr,h), minute(min),second(sec,s),year(yr) が含まれている. 複数形も許されている. date string には date のみ; date と time; date と time と time-zone が含まれている. "year" は時間の単位として使わないことが推奨される. year は長さが変化するので曖昧さがある. Udunits は year を正確に 365 日と定義している.

時間座標変数の認証はその単位によってのみ行なわれる. udunits のルーチン utScan と utIsTime によってこの認証を行なうことができる.

気候的な時間 気候的時間 (12 months, 4 seasons など、特定の年とは無関係なもの) を表す座標変数は、year 0000 で始めること。それ以外は普通の時間軸と同様。

鉛直次元 高さや深さを表す座標変数は常に陽に属性 `units` を含まなくてはならない. デフォルトの属性は用意されていない. 属性 `units` は character タイプである.

鉛直座標として認められる変数は次のもの、およびその複数形。

- (udunits.dat の一覧にあるように) pressure の `units`。最もよく用いられる単位は bar, millibar, decibar, atmosphere (atm)。

- (udunits.dat の一覧にあるように) length の `units`。最もよく用いられる単位は meter(metre,m), centimeter(cm), decimeter(dm), feet(ft)。
- 無次元の `units` である "level" "layer" あるいは "sigma_level"
- 他の udunits.dat の一覧にある単位。ある状況下では, 密度あるいは温度の `units` のような鉛直位置が参照される。

上向きか下向きかの方向性は, 全ての場合に `units` から推定される. これはデータを表示するアプリケーションで役立つ. この理由でこの規約では新しい属性 `positive` を定義する。鉛直軸の `units` が有効な気圧の単位でない場合には属性 `positive` を含めること求められる (udunits ルーチン utScan を使ってこれが決定される)。それ以外の場合にはこの属性はオプションである。属性 `positive` は "up" あるいは"down" (大文字小文字の区別はない) の値をもつ.

鉛直座標変数は次によって認証される.

- 気圧の単位
- "up" あるいは "down" (大文字小文字を区別しない) の値をもつ属性 `positive` の存在

Latitude 次元 latitudes を表現する座標変数は常に陽に属性 `units` を含まなければならない. 属性 `units` のデフォルト値は用意されていない. 属性 `units` は, ユニデータ udunits パッケージで推奨されるフォーマットの character タイプである.

latitude の推奨される単位は "degrees_north" である. "degree_north" "degree_N" "degrees_N" も許されている.

latitude 座標変数の確認は `units` のみによって行なわれる. これは, udunits ルーチン utScan によって行なうことができる.

Longitude 次元 longitude を表現する座標変数は常に陽に属性 `units` を含まなければならない. 属性 `units` のデフォルト値は用意されていない. 属性 `units` は, ユニデータ udunits パッケージで推奨されるフォーマットの character タイプである.

longitude の推奨される単位は "degrees_east" (東向きが正) である. "degree_east" "degree_E" "degrees_E" も許されている. "degrees_west" (西向きが正) は degrees_east からの反転を意味しており, 推奨しない.

longitude は 360 を基本単位として表現される. それで, 例えば -180, 180, 540 は全て日付変更線を表す表現として有効である. また 0 と 360 も prime meridian の表現として有効である. しかし, netCDF ファイル内に格納される longitude の数値は単調でなければならない.

longitude 座標変数の確認は `units` のみによって行なわれる. これは, udunits ルーチン utScan によって行なうことができる.

5 GDT を吟味

ここでは、GDT が独自に定める規約および、COARDS との違いに注目した調査結果を示す。

目的 この規格は気候データの使用を想定しており, 特に GCM で生成されたデータを念頭において作られた. 我々は規格が実際にカバーできる範囲には限界があると認識しており, 気候メタデータのデザインにおいて共通してよく直面すると我々が考えるところの問題に限定している. これは特化した netCDF 規格ではあるが, その考えの多くは広く適用できると思われる. 我々の主目的は気候データに対して必要な明瞭, 適当, かつ柔軟なメタデータの定義を提案することである. メタデータオブジェクトは netCDF 以外のファイルフォーマットに格納できる. 異なったフォーマットのファイル間でメタデータの相互変換は, そのフォーマットが似た考えに基づいているならば容易に行なえる.

この規格は主に COARDS (ftp://ftp.unidata.ucar.edu/pub/netcdf/Conventions/COARDS) によって提供された慣習に追加したものである. 加えて, 特に断わらない限り, 全ての Unidata の推奨がここではサポートされている. 規格間での違いがある部分はコメントで示している.

この規格は, Unidata の提供する udunits 規格も参照している.

ファイル名 COARDS と同様。

データの型 char と byte は機能的に同一であり、**COARDS では char ではなく byte を薦めているが、GDT では byte の sinedness が曖昧との理由で char を薦めている。**

NetCDF は character string 型をサポートしないので、これを char 配列として表現しなければならない。GDT では, これらを string 型と呼ぶ.

属性 この規格が様々な定められた値をとる string の属性を定義する時は, 可能性のある値は小文字で与えられる. しかし, アプリケーションプログラムはこれらの属性の大文字・小文字を区別すべきではない. この規格では, いくつかの string の属性を「空白で分離されたリスト」を含むように定義している. そのリストでは連続する単語は, ひとつ以上の隣接するスペースによって分離される. リストの冒頭あるいは最後には複数のスペースがあってもよい.

グローバル属性 float の属性 `appendices` を用いて GDT の付録のバージョン番号を記録することが推奨される。

string の属性 `quantity_table` は “quantity table(12 章)” の URL の記録に用いられる (12 節)。この属性が null の string の場合には属性 `appendices` が示すバージョンの Appendix D の利用を仮定する。

string の属性 `comment` は、ファイルに関する任意の特別な情報 (例えば GCM の積分名) を記録するのに用いられる。

ユニデータ規格の属性 `history` は、必須ではないが netCDF ファイル内に含まれるデータの進化を記録するために推奨される。

string の属性 `institution` と `production` が推奨される. 属性 `institution` はデータの作成者あるいは提供者を特定する. 属性 `production` はどのようにデータが生成されたか (例えばモデルのバージョン番号や観測手段など) を示す。

属性 calender (23 節) はグローバル属性として記録されることがあり、グローバルな属性 `calendar` は全ての時間軸に対するデフォルトとして解釈される.

変数名 COARDS と同じ

データ変数 物理変数を含むような netCDF 変数をデータ変数 (あるいは primary 変数) と呼ぶ。データ変数の命名規則は特に定めない。

座標変数 ひとつ以上のデータ変数と関連した一次元データ変数を座標変数と呼ぶ。次元名がそれ自身の名前と一致する座標変数名は主要 (main) 座標変数と GDT では呼ぶ。

データ変数の軸と次元 (4 次元以内を強く推奨する **COARDS** とは異なり) データ変数は **0** 次元を含む任意の数の次元をもつことができる。空間あるいは時間以外の軸が含まれてもよい。ひとつの特定の量に 2 つ以上の軸を用意することもできる (28 節)。

格納順は COARDS と同様に TZYX とすべきである (Fortran では X が配列の最初にくる)。空間時間以外の次元が TZYX の左にくることも COARDS と同じ。

もし最後の 4 次元が **(COARDS 規定とは違って)TZYX** 以外の順番をとる場合には、データ変数に属性 `axis` を付けること。それ以外の場合にはこの属性はオプションだが推奨される。この属性はデータ変数の次元と等しいサイズの文字配列で、各次元に対して 1 つの要素をもち、その次元の解釈を与える。許される文字は T, Z, Y, X である。

軸が座標変数をもたない場合には、座標変数は 0 から始まる軸に沿ったインデックスに等しいと仮定される。

(全てのデータ変数がある等圧面で定義されているというような場合に、その情報を含める方法として推奨するのは、) 要素 1 の (気圧) 座標変数をもつ singleton 次元を使うこと。

座標系 属性 axis が X と Y 軸を指し、かつ、これらがそれぞれ経度と緯度である場合には、これらの軸は地球表面に投影された経度-緯度グリッドを構成し、XY-boxes はこの仮定に基づいて計算される。

地理的な球座標ではなく、ある軸に基づく球座標系を "rotated grid" と呼ぶ。"rotated grid" を記述するためには、float の属性 north_pole がデータ変数につけられ、回転した北極に対する (経度、緯度) 座標を特定する。

あるシステムでは、地球表面を被う軸は **rectlinear** グリッドではないが、それでも構わない。でも、その場合の規定は決まっていない。**(COARDS** は非 **rectlinear** 系を認めていない。)

単位 Appendix. C に udunits との違いが書いてある。degrees は認めない。year と month は認めない。calendar_month と calender_year は, お互いあるいは他の時間単位との変換ができない。 ただし、カレンダー年と、12 カレンダー月の倍数との間では変換可能。**COARDS** とはいくつかの点で違いがあるので注意。

変数の物理量 データと座標変数の物理量を特定するための 3 つの string の属性は units, long_name, quantity である。

属性 units は、量が無次元でない場合には必須である。無次元の場合には units は純粋な数字で与えることができる。大・小文字の違いは重要である。

属性 long_name はオプションである。

属性 quantity は、定義されたリスト ("quantity table") から選ばれた記述およびオプション的に ( ) 内に記述された情報によって、その量を明らかにする。リストを定義するのは、出所の異なるデータに対し、どの量が対比できるのかの判断を可能にするため。大・小文字の区別はない。

"quantity talbe" を選択する方法は 2 つある。ひとつは、グローバルな属性 quantity_table を null の string に設定し、この規約の Appendix D のリストを使う方法。もうひとつは同様のリストを web 上に作り、グローバル属性 quantity_table に URL を記録する方法。

属性 subgrid(21 節) は quantity を改良とみなされる。つまり、データ変数のみに適用され、座標変数には適用されない。この 2 つの属性は、共に量の物理次元を定義し、それと矛盾しない *units* をもつ。

データ変数は属性 history をもち、quantity と subgrid に格納できないような、量の導出についての情報を提供する。

データ変数は属性 institution や属性 production をもつこともできる。

属性 history, 属性 instutuion, 属性 production は、ファイル内のデータ変数を区別するために用いるべきではない。

変数に対するオプションの属性 modulo は、量の有効性や物理的重要性を変えることなく加わえたり引いたりできる数を記録する (例えば経度の場合は 360)。

Unidata 標準の属性 FORTRAN_format は、座標とデータ変数の両方に用いられる。

他のモデル依存した属性が、変数の量を定義するために含まれるかもしれない。

軸の形態 周期的な軸は属性 topology="circular" とする。

周期的な軸の main 座標変数は属性 modulo を指定する。

longitude 次元 基本的に COARDS と同じ。ただし、module=360 とすること。**COARDS** では属性 modulo は定義されていないが **modulo=360** が常に仮定されている。地球全体にわたる軸には **COARDS** にはない属性 topology=**"circular"** をつけるべき。

latitude 次元 基本的に COARDS と同じ。

鉛直 (height , depth) の次元 鉛直の軸には属性 long_name と属性 positive (**"up"** または **"down"** を指定) を両方与えること。これは、軸の単位によって正方向を決定する **COARDS** 規定とは異なっている。

Component 変数 連続的な物理変数が、各点における値を特定するために 2 つ以上の数を必要とすることがあり (例えば鉛直 η 座標の場合には σ と p)、これらは components と呼ばれる。components の値は変数内に記録され component 変数と呼ばれる。components が属す変数を components の "head" 変数と呼ぶ。component 変数の名前は head 変数の string の属性 component 内に空白で区切ったリストとして記録される。component 変数の次元は head 変数の次元と一致しなければならない。

座標変数が components を持つ場合には、components を組み合わせて軸上に点を並べるその方法を記した main 座標変数が提供されなければならない。他の場合と同様に、この main 座標変数は単調でなければならないが、components は単調である必要はない。components を使った main 座標の定義は属性 component の ( ) 内で与えられる。この情報は標準化されておらず、そして一般のアプリケーションがこれを使うことは予期されていない。

Associated 変数 データ変数のひとつ以上の軸が、座標値の 2 つ以上の組を持つ場合、これらの組は変数内に記録され、associated 変数と呼ばれ、これら自体が units, long_name, およびその他の適当な属性をもつ。associated 変数の名前は、データ変数あるいは main 座標変数のどちらかの string の属性 associate 内に空白で区切られたリストとして記録される。データ変数との association の場合には、この属性はデータ変数に対してのみ適用される。しかし、main 座標変数との association の場合には、この属性はその main 座標変数を使用する任意のデータ変数に対して適用される。main 座標変数との association は、このようにして一層便利であるが、柔軟性に欠ける。データ変数との association は、複数の軸が含まれており、かつ、main 座標変数がない場合のみのオプションである。

属性 associate は二者択一的かつ等価的に coordinates と名付けられるかもしれない。

変数は 2 つ以上のデータ変数あるいは座標変数と関連しているかもしれない。もし associated 変数自体が属性 associate をもつならば、この属性によって名付けられた変数もまた associated と見なされる。

associated 変数は、それと associated しているデータ変数の全次元をもたなくてはならない。つまり、associated 変数はこれらの軸に対するインデックスの関数とみなすことができる。associated 変数の値は単調である必要はない。

一般のアプリケーションは associated 変数のなんらかの利用を要求されない。associated 変数はデータ変数の属性 axis において指示されない。しかし、ファイルの読み易さを向上させるため、データ変数の属性 associate によって命名された変数が属性 axis における T Z Y または X によって示されると解釈される時は、それ以外の変数の後にこの順番でリストすることが推奨される。

一次元 associated 座標の特別なテクニカルな応用は、ひとつの ultimited 次元に対する netCDF の限界に対処することである。いくつかのデータ変数が長さ或いは物理的重要性が異なる unlimited 軸を持つ場合、それらは全て名目的な unlimited 次元を共有することができ、そして、各々は軸の意味を特定する associated 変数をもつ。

Bundles 同じ物理量を含むいくつかのデータ配列が 1 つ以上の同じ軸を持ち、しかし他の singleton 座標変数の値によって区別される場合に、属性 associate を使うことでそれらを同じデータ変数内に格納することができる。

境界変数 ある次元に沿って、値が点あるいは、連続または不連続なセルと関連しているとする。点の座標値の境界値および、セルの座標値の境界値を定義しなければならない。この境界値は string の属性 bounds によって記録される。その際に推奨される名前は、"bounds_座標の次元" である。valid_min が小さい境界値と等しい場合、その区間は小さい境界値を持たない。valid_max が大きい境界値と等しい場合、その区間は大きい境界値を持たない。

subgrid 変数の表現 データ変数の属性 subgrid を用い、ある軸に沿って各データの値がどの程度格子以下の変動を反映しているか示すことができる。これは string の属性であり、空白で区切られた単語のリストからなる。リスト内の"name:method" は "name" が与えられた次元の軸に沿っての格子点以下の変動が特定された "method" によって表現されることを示す。method は Appendix B に示された項目の中のひとつであり、mean, maximum, minimum, mid-range, standard deviation, variance, mode, median, cell, point を含んでいる。大・小文字や句読点は重要ではない。

2 つ以上の subgrid method が指示される場合には、それが適用される順に左から並べる。

縮約次元 縮約軸とは、大きな dimension (要素の数) をもつ軸の値をまとめて最初よりも少ないグループにすることで作られた軸である。もっともありふれた場合には、dimension は完全に singleton 次元に圧縮され、全てのデータ点は完全に圧縮された軸を共有する。

縮約される前の軸情報を記録するオプション属性として、`old_interval` (縮約されていない 2 つの近接する座標間の典型的な距離) と `old_spacing`("uniform" の場合は `old_interval` が等間隔, "variable" の場合は不等間隔だが近接している, "disjoint" の場合はギャップがある) がある。縮約される前の軸の座標値がその変数内で陽に記録されることがある。その場合には、縮約される前の main 座標変数は縮約された main 座標変数の属性 `expand` によって命名されるべきである。

時間変数と間隔 時間変数 (以下時間と呼ぶ) は日時をあらわす変数。時間間隔は 2 つの時間の差。時間は 6 つの要素 (year, month, day, hour, minutes, second) で記述することができる。これらの要素を変換してただひとつの数として表すことをエンコードといい、その逆をデコードという。これらはカレンダーに強く依存する。

属性 `calendar` は、使用するカレンダーを指定する。もし時間座標変数が属性 `calendar` をもたないなら、(もしあれば) グローバル属性 `calendar` が適用される。

規約では、6 つの要素で表現される時間をひとつの数にエンコードする 2 つの方法が許されており、これらは `units` で区別される。ひとつは相対時間へのエンコード、もうひとつは絶対時間へのエンコードである (24,25 節)。

時間変数は属性 `time_format` をもつことがあり、日時をプリントするフォーマットを UNIX の date コマンドのコンベンションによって特定する。

時間座標変数は、常に陽に属性 `units` を含まなければならない。デフォルトの値はない。

相対時間 特定の標準時間から経過した時間。`units` は "時間単位 since 標準時間" のようになる。デコードするためにはカレンダーが必要。

udunits.dat ファイルは秒、分、時間、日を時間の単位として定義している。月と年はこの規格の Appendix C では認めていない。これは、それらがうまく定義されていないため。udunits の単位 common_year (正確に 365 日) は使ってもよいが、推奨はしない。

絶対時間 属性 `units` は "時間単位 as 時間 string " となる。可能な単位と推奨されるデータタイプおよび意味はつぎのとおり:

Format	Data type	Interpretation
second as %S.%f	float	Diurnal phase
minute as %M.%f	float	Diurnal phase
hour as %H.%f	float	Diurnal phase
day as %Y%m%d.%f	double	Time
day as %Y%m%d	int	Year and seasonal phase
day as %m%d.%f	double	Seasonal phase and diurnal phase
day as %m%d	int	Seasonal phase
day as .%f	float	Diurnal phase
calendar_month as %Y%m.%f	double	Year and seasonal phase
calendar_month as %m.%f	float	Seasonal phase
calendar_year as %Y.%f	double	Year and seasonal phase
calendar_year as %Y	int	Year
calendar_year as .%f	float	Seasonal phase

calendar_year と calendar_month はこの規約の Appendix C で定義された時間の単位。

時間 string コードは UNIX の date と printf コマンドに従っている。

Format letter	Interpretation
%Y	Year (including century)
%m	Two-digit month (01=January)
%d	Two-digit day within month
%H	Hours since midnight
%M	Minutes since midnight
%S	Seconds since midnight
%f	Floating-point fraction of the specified time-unit
.	Position of decimal point

グレゴリオ暦 絶対時間を扱えないアプリケーションとの適合性が本質でないならば、グレゴリオ時間を、データタイプ は double で、`units` は "day as %Y%m%d.%f" で与えることを推奨する。それが本質である場合には、相対時間とする。推奨する単位は days、データタイプは double。

時間変数に適用する属性 `calendar` がない場合には、値は通常のグレゴリオ暦であると仮定される。これは `calendar` に standard または gregorian を設定することで陽に指定できる。

非グレゴリオ暦 他のカレンダータイプの場合には、データタイプは double で、`units` は "day as %Y%m%d.%f" とすることを推奨する。相対時間は認められており、推奨される `units` は days since 1-1-1 (1 年 1 月 1 日の真夜中) であり、データタイプは double である。Unidata の udunits パッケージは標準カレンダーだけを処理できるので、他のカレンダーに対して相対時間を処理するための拡張が必要となる。

グレゴリオ暦以外で、この規約が認識するカレンダーは julian (ユリウス暦)、noleap (各月が 30 日で一年が 360 日) である。それ以外のカレンダーを使うときは、属性 `calendar` に適当に記述すべきであるが、一般のアプリケーションがこれを処理することは予期されていない。

多重時間軸と気候的時間 もし、collapsed 軸と結びついた uncollapsed な軸があれば、それは "climatological time" 軸である。

CAORDS とは異なり、気候的時間を **year 0** から始める必要はない。

データ変数における無効値 基本的にドキュメントの `valid_min`, `valid_max`, `valid_range` と同じ

データ変数における欠損値 ドキュメントの missgin_value の項目を見よ。

gathering による圧縮 データ配列から常に欠損となる点を消去したいことがある。そのような圧縮は隣りあったひとつ以上の軸にわたって操作でき、記憶されるべき点のリストを参照することで達成できる。このリストは単に圧縮される軸をもつマスク配列を考慮し、ならべ替えなしにこの配列を一次元上にマッピングすることで、作成される。リストは、要求された点のこの一次元マスクにおけるインデックスの集合である。圧縮された行列では、圧縮されるべき軸は全て、次元が要求された点の数であるただひとつの軸に取り換えられる。要求された点は、この軸に沿って、圧縮されていない配列で表れた順番で、要求されていない点を飛ばして現れる。圧縮および戻す操作はこのリストにわたってループの操作をすることで実行される。

このリストは、データ配列の圧縮された軸に対する座標変数として記憶される。こうして、リスト変数とその次元は同じ名前をもつ。このリスト変数は string の属性 `compress` をもち、圧縮されていない配列のファイル内での宣言の順番で、圧縮によって影響される次元の空白で区切られたリストを含む。この属性の存在によって、リスト変数がそのようなものだと確認される。リスト、オリジナルの次元と座標変数 (component, associated, boundary 変数を含む)、および、圧縮されない変数の全ての属性をもつ圧縮されたデータ変数が、netCDF ファイルアーカイブに書かれる。圧縮されていない変数は、そのオリジナル変数名が知られていない場合を除いて、この情報を使って正確にもとどおりに構成されることになる。

scale と **offset** を用いた圧縮 COARDS の `add_offset` の内容と基本的に同じ。また、ドキュメントの `add_offset` を見よ。ただし、`missing_value` と `_FillValue` の扱いに関連して **COARDS** と若干の違いがある (ドキュメントの `missing_value` の項目を見よ)。

6 属性一覧表

属性	doc	coa-rds	gdt	種	Use	GDT の節	Description
add_offset	○	○	○	N	CD	29, 32	Additive offset for packing data
appendices			○	S	G	5	Version number of these appendices
associate			○	S	CD	18, 19	Identifies variables containing alternative sets of coordinates
axis			○ [1]	S	D	9, 16, 18	Identifies spatiotemporal dimensions
bounds			○	N	C	20, 22, 28	Identifies a bounday variable
calendar			○	S	GD	5, 23, 26, 27	Calendar used for encoding time axes
comment			○	S	G	5	Additional information about the file
component			○	S	CD	17, 20	Identifies variables containing components of a variable
compress			○	S	D	31	Records dimensions which have been compressed by gathering
Conventions	○	○	○	S	G	5	Identifies the netCDF standard
coordinates			○	S	CD	18	Synonym for associate
expand			○	S	C	22, 28	Coordinates before contraction
_FillValue	○	△	△ [4]	N	D	29	Indicator of invalid data
FORTRAN_format	○		○	S	CD	12	Format for a variable
history	○	○	○	S	GD	5, 12	Evolution of the data in the file
institution			○	S	GD	5, 12	Who made or supplied the data
long_name	○	○	○	S	CD	12	Long description of a physical quantity
modulo			○	N	CD	12, 14, 25	Arithmetic modulo of a variable
north_pole			○	N	D	10	Long.,lat. of rotated North Pole
old_interval			○	N	C	22, 28	The typical separation between points on an axis before contraction
old_spacing			○	S	C	22, 28	Indicates the spacing of points along an axis before contraction
positive		△	△ [3]	S	C	16	Direction of positive for a vertical axis
production			○	S	GD	5, 12	How the data was produced
quantity			○	S	CD	12	Standardised description of a physical quantity
quantity_table			○	S	G	5, 12	URL of the quantity table
scale_factor	○	○	○	N	CD	29, 32	Multiplicative factor for packing data
subgrid			○	S	D	21, 22, 28	Records how the data values represent subgrid variation
topology			○	S	C	13, 14, 25	Topology of an axis (circular or not)
time_format			○	S	CD	23	Format for printing a time and date
units	△	△	△ [2]	S	CD	12, 14, 15, 23 - 27	Units of a physical quantity
valid_max	○		○	N	CD	20, 29	Largest valid value of a variable
valid_min	○		○	N	CD	20, 29	Smallest valid value of a variable
valid_range	○		○	N	CD	29	Smallest and largest valid values of a variable
missing_value	○	○	○	N	D	30	a value for data that are unknown or missing
signedness	○			?	D?		byte 値の符号の扱いを指示。事実上不要
C_format	○			S	CD		Format for a variable
title	○			S	G		データセットの中身を簡単に記述

S は string、N は数値。G はグローバル、C は座標変数 (多次元座標変数を含む) 関連、D はデータ変数関連。

○は説明があるもの。△は説明があるが他の規約と相異点があるもの。

[1] この結果、軸の格納の順番が COARDS よりも柔軟になる。
[2] 基本的に udunits に従うが、いろいろと例外があるので注意。
[3] COARDS と GDT とではこの属性を利用するルールが異なる．これと関連して `units` の指定方法も異なる。
[4] COARDS では、データが属性 `scalar_factor` と属性 `add_offset` を使ってパックされる際に、`missing_value` も `_FillValue` も同様にパックされる。しかし、GDT では `_FillValue` の値はパックされないので、アンパックする再にはあらかじめ `_FillValue` の存在を確認しておくこと。`missing_value` はパックされるので、一緒にアンパックする。