Package 'RNetCDF'

February 21, 2016

Version 1.8-2

Date 2016-02-21

Title Interface to NetCDF Datasets

Author Pavel Michna, with contributions from Milton Woods

Maintainer Milton Woods <mwoods@users.r-forge.r-project.org>

Depends R (>= 2.5.0)

SystemRequirements netcdf (>= 3.6.0), udunits (>= 1.11.7) or udunits2 (>= 2.1.22)

Description An interface to the NetCDF file format designed by Unidata for efficient storage of array-oriented scientific data and descriptions. The R interface is closely based on the C API of the NetCDF library, and it includes calendar conversions from the Unidata UDUNITS library. The current implementation supports all operations on NetCDF datasets in classic and 64-bit offset file formats, and NetCDF4-classic format is supported for reading and modification of existing files.

License GPL (>= 2) | file LICENSE

URL http://rnetcdf.r-forge.r-project.org
http://www.unidata.ucar.edu/software/netcdf/
http://www.unidata.ucar.edu/software/udunits/

NeedsCompilation yes

Repository CRAN

Repository/R-Forge/Project rnetcdf

Repository/R-Forge/Revision 76

Repository/R-Forge/DateTimeStamp 2016-02-21 00:07:03

Date/Publication 2016-02-21 09:02:40

R topics documented:

`att.copy.nc` *Copy Attribute from One NetCDF to Another*

Description

Copy attribute from one NetCDF to another.

Usage

```
att.copy.nc(ncfile.in, variable.in, attribute, ncfile.out, variable.out)
```

Arguments

`ncfile.in`	Object of class `"NetCDF"` which points to the input NetCDF dataset from which the attribute will be copied (as returned from `open.nc`).
`variable.in`	ID or name of the variable in the input NetCDF dataset from which the attribute will be copied, or `"NC_GLOBAL"` for a global attribute.
`attribute`	Name or ID of the attribute in the input NetCDF dataset to be copied.
`ncfile.out`	Object of class `"NetCDF"` which points to the output NetCDF dataset to which the attribute will be copied (as returned from `open.nc`). It is permissible for the input and output NetCDF object to be the same.

`variable.out` ID or name of the variable in the output NetCDF dataset to which the attribute will be copied, or `"NC_GLOBAL"` to copy to a global attribute.

Details

This function copies an attribute from one open NetCDF dataset to another. It can also be used to copy an attribute from one variable to another within the same NetCDF dataset.

Author(s)

Pavel Michna

References

http://www.unidata.ucar.edu/software/netcdf/

Examples

```
##  Create two new NetCDF datasets and define two dimensions
nc.1 <- create.nc("foo_1.nc")
nc.2 <- create.nc("foo_2.nc")

dim.def.nc(nc.1, "station", 5)
dim.def.nc(nc.1, "time", unlim=TRUE)

dim.def.nc(nc.2, "station", 5)
dim.def.nc(nc.2, "time", unlim=TRUE)

##  Create two variables, one as coordinate variable
var.def.nc(nc.1, "time", "NC_INT", "time")
var.def.nc(nc.1, "temperature", "NC_DOUBLE", c(0,1))

var.def.nc(nc.2, "time", "NC_INT", "time")
var.def.nc(nc.2, "temperature", "NC_DOUBLE", c(0,1))

##  Put some attributes to the first dataset
att.put.nc(nc.1, "temperature", "missing_value", "NC_DOUBLE", -99999.9)
att.put.nc(nc.1, "NC_GLOBAL", "title", "NC_CHAR", "Data from Foo")

##  Copy the attributes to the second dataset
att.copy.nc(nc.1, 1, 0, nc.2, 1)
att.copy.nc(nc.1, "NC_GLOBAL", "title", nc.2, "NC_GLOBAL")

close.nc(nc.1)
close.nc(nc.2)
```

att.delete.nc *Delete a NetCDF Attribute*

Description

Delete a NetCDF attribute.

Usage

```
att.delete.nc(ncfile, variable, attribute)
```

Arguments

ncfile	Object of class "NetCDF" which points to the NetCDF dataset (as returned from open.nc).
variable	ID or name of the attribute's variable, or "NC_GLOBAL" for a global attribute.
attribute	The name of the attribute to be deleted.

Details

This function deletes a NetCDF attribute from a NetCDF dataset open for writing.

Author(s)

Pavel Michna

References

http://www.unidata.ucar.edu/software/netcdf/

Examples

```
##  Create a new NetCDF dataset and define two dimensions
nc <- create.nc("foo.nc")

dim.def.nc(nc, "station", 5)
dim.def.nc(nc, "time", unlim=TRUE)

##  Create two variables, one as coordinate variable
var.def.nc(nc, "time", "NC_INT", "time")
var.def.nc(nc, "temperature", "NC_DOUBLE", c(0,1))

##  Put some attributes
att.put.nc(nc, "temperature", "missing_value", "NC_DOUBLE", -99999.9)
att.put.nc(nc, "NC_GLOBAL", "title", "NC_CHAR", "Data from Foo")

##  Delete these attributes
att.delete.nc(nc, "temperature", "missing_value")
att.delete.nc(nc, "NC_GLOBAL", "title")
```

```
close.nc(nc)
```

`att.get.nc` *Get a NetCDF Attribute*

Description

Get an attribute from a NetCDF dataset.

Usage

```
att.get.nc(ncfile, variable, attribute)
```

Arguments

`ncfile`	Object of class `"NetCDF"` which points to the NetCDF dataset (as returned from `open.nc`).
`variable`	ID or name of the variable from which the attribute will be read, or `"NC_GLOBAL"` for a global attribute.
`attribute`	Attribute name or ID.

Details

This function returns the value of the attribute.

Value

A vector of type `character` if the on-disk type is `NC_CHAR`, otherwise `numeric`. No distinction is made between the different storage types of numeric objects.

Note

`NC_BYTE` is always interpreted as signed.

Author(s)

Pavel Michna

References

http://www.unidata.ucar.edu/software/netcdf/

Examples

```
##  Create a new NetCDF dataset and define two dimensions
nc <- create.nc("foo.nc")

dim.def.nc(nc, "station", 5)
dim.def.nc(nc, "time", unlim=TRUE)

##  Create two variables, one as coordinate variable
var.def.nc(nc, "time", "NC_INT", "time")
var.def.nc(nc, "temperature", "NC_DOUBLE", c(0,1))

##  Put some attributes
att.put.nc(nc, "temperature", "missing_value", "NC_DOUBLE", -99999.9)
att.put.nc(nc, "temperature", "long_name", "NC_CHAR", "air temperature")
att.put.nc(nc, "NC_GLOBAL", "title", "NC_CHAR", "Data from Foo")
att.put.nc(nc, "NC_GLOBAL", "history", "NC_CHAR", paste("Created on", date()))

##  Get these attributes
att.get.nc(nc, "temperature", "missing_value")
att.get.nc(nc, "temperature", "long_name")
att.get.nc(nc, "NC_GLOBAL", "title")
att.get.nc(nc, "NC_GLOBAL", "history")

close.nc(nc)
```

---

att.inq.nc *Inquire About a NetCDF Attribute*

Description

Inquire about a NetCDF attribute.

Usage

```
att.inq.nc(ncfile, variable, attribute)
```

Arguments

ncfile	Object of class "NetCDF" which points to the NetCDF dataset (as returned from open.nc).
variable	Either the ID or the name of the attribute's variable or "NC_GLOBAL" for a global attribute.
attribute	Either the ID or the name of the attribute to be inquired.

Details

This function returns information about a NetCDF attribute. Information about an attribute include its ID, its name, its type, and its length. The valid external NetCDF data types are NC_BYTE, NC_CHAR, NC_SHORT, NC_INT, NC_FLOAT, and NC_DOUBLE. In general, attributes are rather accessed by name than by their ID (which is called number) because the attribute number is more volatile than the name, since it can change when other attributes of the same variable are deleted.

Value

A list containing the following components:

id	Attribute ID.
name	Attribute name.
type	External NetCDF data type.
length	Length of this attribute.

Author(s)

Pavel Michna

References

http://www.unidata.ucar.edu/software/netcdf/

Examples

```
##  Create a new NetCDF dataset and define two dimensions
nc <- create.nc("foo.nc")

dim.def.nc(nc, "station", 5)
dim.def.nc(nc, "time", unlim=TRUE)

##  Create two variables, one as coordinate variable
var.def.nc(nc, "time", "NC_INT", "time")
var.def.nc(nc, "temperature", "NC_DOUBLE", c(0,1))

##  Put some attributes
att.put.nc(nc, "temperature", "missing_value", "NC_DOUBLE", -99999.9)
att.put.nc(nc, "NC_GLOBAL", "title", "NC_CHAR", "Data from Foo")

##  Inquire about these attributes
att.inq.nc(nc, "temperature", "missing_value")
att.inq.nc(nc, "NC_GLOBAL", "title")

close.nc(nc)
```

`att.put.nc` *Put a NetCDF Attribute*

Description

Put an attribute to a NetCDF dataset.

Usage

```
att.put.nc(ncfile, variable, name, type, value)
```

Arguments

`ncfile`	Object of class `"NetCDF"` which points to the NetCDF dataset (as returned from `open.nc`).
`variable`	ID or name of the variable to which the attribute will be assigned or `"NC_GLOBAL"` for a global attribute.
`name`	Attribute name. Must begin with an alphabetic character, followed by zero or more alphanumeric characters including the underscore (`"_"`). Case is significant. Attribute name conventions are assumed by some NetCDF generic applications, e.g., `units` as the name for a string attribute that gives the units for a NetCDF variable.
`type`	One of the set of predefined NetCDF external data types. The valid NetCDF external data types are `NC_BYTE`, `NC_CHAR`, `NC_SHORT`, `NC_INT`, `NC_FLOAT`, and `NC_DOUBLE`.
`value`	Attribute value. This can be either a single numeric value or a vector of numeric values, or alternatively a character string.

Details

Names commencing with underscore (`"_"`) are reserved for use by the NetCDF library. Most generic applications that process NetCDF datasets assume standard attribute conventions and it is strongly recommended that these be followed unless there are good reasons for not doing so.

Note

`NC_BYTE` is always interpreted as signed.

Author(s)

Pavel Michna

References

http://www.unidata.ucar.edu/software/netcdf/

Examples

```
##  Create a new NetCDF dataset and define two dimensions
nc <- create.nc("foo.nc")

dim.def.nc(nc, "station", 5)
dim.def.nc(nc, "time", unlim=TRUE)

##  Create two variables, one as coordinate variable
var.def.nc(nc, "time", "NC_INT", "time")
var.def.nc(nc, "temperature", "NC_DOUBLE", c(0,1))

##  Put some attributes
att.put.nc(nc, "temperature", "missing_value", "NC_DOUBLE", -99999.9)
att.put.nc(nc, "temperature", "long_name", "NC_CHAR", "air temperature")
att.put.nc(nc, "NC_GLOBAL", "title", "NC_CHAR", "Data from Foo")
att.put.nc(nc, "NC_GLOBAL", "history", "NC_CHAR", paste("Created on", date()))

close.nc(nc)
```

`att.rename.nc` *Rename a NetCDF Attribute*

Description

Rename a NetCDF attribute.

Usage

```
att.rename.nc(ncfile, variable, attribute, newname)
```

Arguments

`ncfile`	Object of class `"NetCDF"` which points to the NetCDF dataset (as returned from `open.nc`).
`variable`	ID or name of the attribute's variable, or `"NC_GLOBAL"` for a global attribute.
`attribute`	The current attribute name or ID.
`newname`	The new name to be assigned to the specified attribute.

Details

This function changes the name of an existing attribute in a NetCDF dataset open for writing. An attribute cannot be renamed to have the same name as another attribute of the same variable.

Author(s)

Pavel Michna

References

http://www.unidata.ucar.edu/software/netcdf/

Examples

```
##  Create a new NetCDF dataset and define two dimensions
nc <- create.nc("foo.nc")

dim.def.nc(nc, "station", 5)
dim.def.nc(nc, "time", unlim=TRUE)

##  Create two variables, one as coordinate variable
var.def.nc(nc, "time", "NC_INT", "time")
var.def.nc(nc, "temperature", "NC_DOUBLE", c(0,1))

##  Put some attributes
att.put.nc(nc, "temperature", "missing_value", "NC_DOUBLE", -99999.9)
att.put.nc(nc, "NC_GLOBAL", "title", "NC_CHAR", "Data from Foo")

##  Rename these attributes
att.rename.nc(nc, "temperature", "missing_value", "my_missing_value")
att.rename.nc(nc, "NC_GLOBAL", "title", "my_title")

close.nc(nc)
```

---

close.nc *Close a NetCDF Dataset*

Description

Close an open NetCDF dataset.

Usage

```
close.nc(con, ...)
```

Arguments

con	Object of class "NetCDF" which points to the NetCDF dataset (as returned from open.nc).
...	Arguments passed to or from other methods (not used).

Details

This function closes an open NetCDF dataset. After an open NetCDF dataset is closed, its NetCDF ID may be reassigned to the next NetCDF dataset that is opened or created. Therefore, the passed object (ncfile) should be deleted by the user after calling this function.

Author(s)

Pavel Michna

References

http://www.unidata.ucar.edu/software/netcdf/

Examples

```
##  Create a void NetCDF dataset
nc <- create.nc("foo.nc")
close.nc(nc)
```

---

create.nc *Create a NetCDF Dataset*

Description

Create a new NetCDF dataset.

Usage

```
create.nc(filename, clobber=TRUE, large=FALSE, share=FALSE, prefill=TRUE)
```

Arguments

filename	Filename for the NetCDF dataset to be created.
clobber	The creation mode. If TRUE (default), any existing dataset with the same filename will be overwritten. Otherwise set to FALSE.
large	The file format. If FALSE (default), create a NetCDF classic format file, otherwise create a 64-bit offset format file. The 64-bit offset format imposes fewer restrictions on data files larger than 2 GB, but it cannot be read by NetCDF library versions earlier than 3.6.0.
share	The buffer scheme. If FALSE (default), dataset access is buffered and cached for performance. However, if one or more processes may be reading while another process is writing the dataset, set to FALSE.
prefill	The prefill mode. If TRUE (default), newly defined variables are initialised with fill values when they are first accessed. This allows unwritten array elements to be detected when reading, but it also implies duplicate writes if all elements are subsequently written with user-specified data. Enhanced write performance can be obtained by setting prefill=FALSE.

Details

This function creates a new NetCDF dataset, returning an object of class "NetCDF" that can be used in R. A creation mode flag specifies whether to overwrite any existing dataset with the same name.

Value

Object of class "NetCDF" which points to the NetCDF dataset.

Author(s)

Pavel Michna, Milton Woods

References

http://www.unidata.ucar.edu/software/netcdf/

Examples

```
##  Create a void NetCDF dataset
nc <- create.nc("foo.nc")
close.nc(nc)
```

---

dim.def.nc *Define a NetCDF Dimension*

Description

Define a new NetCDF dimension.

Usage

```
dim.def.nc(ncfile, dimname, dimlength=1, unlim=FALSE)
```

Arguments

ncfile	Object of class "NetCDF" which points to the NetCDF dataset (as returned from open.nc).
dimname	Dimension name. Must begin with an alphabetic character, followed by zero or more alphanumeric characters including the underscore ("_"). Case is significant.
dimlength	Length of dimension, that is, number of values for this dimension as an index to variables that use it. This must be a positive integer. If an unlimited dimension is created (unlim=TRUE), the value of length is not used.
unlim	Set to TRUE if an unlimited dimension should be created, otherwise to FALSE.

Details

This function creates a new NetCDF dimension. There is a suggested limit (100) to the number of dimensions. Ordinarily, the name and length of a dimension are fixed when the dimension is first defined. The name may be changed later, but the length of a dimension (other than the unlimited dimension) cannot be changed without copying all the data to a new NetCDF dataset with a redefined dimension length. A NetCDF dimension in an open NetCDF dataset is referred to by a small integer called a dimension ID. In the C interface, dimension IDs are 0, 1, 2, ..., in the order in which the dimensions were defined. At most one unlimited length dimension may be defined for each NetCDF dataset.

Author(s)

Pavel Michna

References

http://www.unidata.ucar.edu/software/netcdf/

Examples

```
##  Create a new NetCDF dataset and define two dimensions
nc <- create.nc("foo.nc")

dim.def.nc(nc, "station", 5)
dim.def.nc(nc, "time", unlim=TRUE)

close.nc(nc)
```

---

dim.inq.nc *Inquire About a NetCDF Dimension*

Description

Inquire about a NetCDF dimension.

Usage

```
dim.inq.nc(ncfile, dimension)
```

Arguments

ncfile	Object of class "NetCDF" which points to the NetCDF dataset (as returned from open.nc).
dimension	Either the ID or the name of the dimension to be inquired.

Details

This function returns information about a NetCDF dimension. Information about a dimension include its name, its ID, its length and a flag if it is the unlimited dimension of this NetCDF dataset, if any. The length of the unlimited dimension, if any, is the number of records written so far.

Value

A list containing the following components:

`id`	Dimension ID.
`name`	Dimension name.
`length`	Length of dimension. For the unlimited dimension, this is the number of records written so far.
`unlim`	`TRUE` if it is the unlimited dimension, `FALSE` otherwise.

Author(s)

Pavel Michna

References

http://www.unidata.ucar.edu/software/netcdf/

Examples

```
##  Create a new NetCDF dataset and define two dimensions
nc <- create.nc("foo.nc")

dim.def.nc(nc, "station", 5)
dim.def.nc(nc, "time", unlim=TRUE)

##  Inquire about the dimensions
dim.inq.nc(nc, 0)
dim.inq.nc(nc, "time")

close.nc(nc)
```

---

dim.rename.nc *Rename a NetCDF Dimension*

Description

Rename a NetCDF dimension.

Usage

```
dim.rename.nc(ncfile, dimension, newname)
```

Arguments

ncfile	Object of class "NetCDF" which points to the NetCDF dataset (as returned from open.nc).
dimension	Either the ID or the name of the dimension to be renamed.
newname	The new dimension name.

Details

This function renames an existing dimension in a NetCDF dataset open for writing. A dimension cannot be renamed to have the same name as another dimension.

Author(s)

Pavel Michna

References

http://www.unidata.ucar.edu/software/netcdf/

Examples

```
##  Create a new NetCDF dataset and define two dimensions
nc <- create.nc("foo.nc")

dim.def.nc(nc, "station", 5)
dim.def.nc(nc, "time", unlim=TRUE)

##  Rename the dimensions
dim.rename.nc(nc, 0, "mystation")
dim.rename.nc(nc, "time", "mytime")

close.nc(nc)
```

---

file.inq.nc *Inquire About a NetCDF Dataset*

Description

Inquire about a NetCDF dataset.

Usage

```
file.inq.nc(ncfile)
```

Arguments

ncfile	Object of class "NetCDF" which points to the NetCDF dataset (as returned from open.nc).

Details

This function returns values for the number of dimensions, the number of variables, the number of global attributes, and the dimension ID of the dimension defined with unlimited length, if any.

Value

A list containing the following components:

ndims	Number of dimensions defined for this NetCDF dataset.
nvars	Number of variables defined for this NetCDF dataset.
ngatts	Number of global attributes for this NetCDF dataset.
unlimdimid	ID of the unlimited dimension, if there is one for this NetCDF dataset. Otherwise NA will be returned.

Author(s)

Pavel Michna

References

http://www.unidata.ucar.edu/software/netcdf/

Examples

```
##  Create a new NetCDF dataset and define two dimensions
nc <- create.nc("foo.nc")

dim.def.nc(nc, "station", 5)
dim.def.nc(nc, "time", unlim=TRUE)

##  Create two variables, one as coordinate variable
var.def.nc(nc, "time", "NC_INT", "time")
var.def.nc(nc, "temperature", "NC_DOUBLE", c(0,1))

##  Put some attributes
att.put.nc(nc, "temperature", "missing_value", "NC_DOUBLE", -99999.9)
att.put.nc(nc, "temperature", "long_name", "NC_CHAR", "air temperature")
att.put.nc(nc, "NC_GLOBAL", "title", "NC_CHAR", "Data from Foo")
att.put.nc(nc, "NC_GLOBAL", "history", "NC_CHAR", paste("Created on", date()))

##  Inquire about the dataset
file.inq.nc(nc)

close.nc(nc)
```

open.nc *Open a NetCDF Dataset*

Description

Open an existing NetCDF dataset for reading and (optionally) writing.

Usage

```
open.nc(con, write=FALSE, share=FALSE, prefill=TRUE, ...)
```

Arguments

con	Filename of the NetCDF dataset to be opened.
write	If FALSE (default), the dataset will be opened read-only. If TRUE, the dataset will be opened read-write.
share	The buffer scheme. If FALSE (default), dataset access is buffered and cached for performance. However, if one or more processes may be reading while another process is writing the dataset, set to FALSE.
prefill	The prefill mode. If TRUE (default), newly defined variables are initialised with fill values when they are first accessed. This allows unwritten array elements to be detected when reading, but it also implies duplicate writes if all elements are subsequently written with user-specified data. Enhanced write performance can be obtained by setting prefill=FALSE.
...	Arguments passed to or from other methods (not used).

Details

This function opens an existing NetCDF dataset for access. By default, the dataset is opened read-only. If write=TRUE, then the dataset can be changed. This includes appending or changing data, adding dimensions, variables, and attributes.

Value

Object of class "NetCDF" which points to the NetCDF dataset.

Author(s)

Pavel Michna, Milton Woods

References

http://www.unidata.ucar.edu/software/netcdf/

Examples

```
##  Create a void NetCDF dataset
nc <- create.nc("foo.nc")
close.nc(nc)

##  Open the NetCDF dataset for writing
nc <- open.nc("foo.nc", write=TRUE)
close.nc(nc)
```

`print.nc` *Print Summary Information About a NetCDF Dataset*

Description

Print summary information about a NetCDF dataset.

Usage

```
print.nc(x, ...)
```

Arguments

`x`	Object of class `"NetCDF"` which points to the NetCDF dataset (as returned from `open.nc`).
`...`	Arguments passed to or from other methods (not used).

Details

This function prints information about a NetCDF dataset. This includes a list of all dimensions and their length, a list of all variables and their attributes (including their values) and a list of all global attributes (including their values).

The output of this function is almost identical with a `"ncdump -h"` call. Because arrays in R have their leftmost subscript varying fastest, the fastest varying dimensions are printed first.

Author(s)

Pavel Michna

References

http://www.unidata.ucar.edu/software/netcdf/

Examples

```
##  Create a new NetCDF dataset and define two dimensions
nc <- create.nc("foo.nc")

dim.def.nc(nc, "station", 5)
dim.def.nc(nc, "time", unlim=TRUE)

##  Create two variables, one as coordinate variable
var.def.nc(nc, "time", "NC_INT", "time")
var.def.nc(nc, "temperature", "NC_DOUBLE", c(0,1))

##  Put some attributes
att.put.nc(nc, "temperature", "missing_value", "NC_DOUBLE", -99999.9)
att.put.nc(nc, "temperature", "long_name", "NC_CHAR", "air temperature")
att.put.nc(nc, "NC_GLOBAL", "title", "NC_CHAR", "Data from Foo")
att.put.nc(nc, "NC_GLOBAL", "history", "NC_CHAR", paste("Created on", date()))

##  Print summary information about the dataset
print.nc(nc)

close.nc(nc)
```

---

read.nc *Read a NetCDF Dataset*

Description

Read all data from a NetCDF dataset.

Usage

```
read.nc(ncfile, unpack=TRUE)
```

Arguments

ncfile	Object of class "NetCDF" which points to the NetCDF dataset (as returned from open.nc).
unpack	Unpack "packed" variables if set to TRUE (default).

Details

This function reads all variable data from a NetCDF dataset into a single list. The list elements (arrays) have the same names as the variables in the NetCDF dataset.

Packed variables can optionally be returned in an unpacked state (see var.get.nc for more information).

Value

A list with the list elements containing the variable data of the NetCDF dataset.

Author(s)

Pavel Michna, Milton Woods

References

http://www.unidata.ucar.edu/software/netcdf/

Examples

```
##  Create a new NetCDF dataset and define two dimensions
nc <- create.nc("foo.nc")

dim.def.nc(nc, "station", 5)
dim.def.nc(nc, "time", unlim=TRUE)
dim.def.nc(nc, "max_string_length", 32)

##  Create three variables, one as coordinate variable
var.def.nc(nc, "time", "NC_INT", "time")
var.def.nc(nc, "temperature", "NC_DOUBLE", c(0,1))
var.def.nc(nc, "name", "NC_CHAR", c("max_string_length", "station"))

##  Put some missing_value attribute for temperature
att.put.nc(nc, "temperature", "missing_value", "NC_DOUBLE", -99999.9)

##  Define variable values
mytime        <- c(1:2)
mytemperature <- c(1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, NA, NA, 9.9)
myname        <- c("alfa", "bravo", "charlie", "delta", "echo")

##  Put the data
var.put.nc(nc, "time", mytime, 1, length(mytime))
var.put.nc(nc, "temperature", mytemperature, c(1,1), c(5,2))
var.put.nc(nc, "name", myname, c(1,1), c(32,5))

sync.nc(nc)

##  Get the data
read.nc(nc)

close.nc(nc)
```

RNetCDF *R Interface to NetCDF Datasets*

Description

This package provides an interface to Unidata's NetCDF library functions (version 3) and furthermore access to Unidata's UDUNITS calendar conversions. The routines and the documentation

follow the NetCDF and UDUNITS C interface, so the corresponding manuals can be consulted for more detailed information.

NetCDF is an abstraction that supports a view of data as a collection of self-describing, portable objects that can be accessed through a simple interface. Array values may be accessed directly, without knowing details of how the data are stored. Auxiliary information about the data, such as what units are used, may be stored with the data. Generic utilities and application programs can access NetCDF datasets and transform, combine, analyze, or display specified fields of the data.

The external types supported by the NetCDF interface are:

NC_CHAR	8-bit characters intended for representing text.
NC_BYTE	8-bit signed or unsigned integers.
NC_SHORT	16-bit signed integers.
NC_INT	32-bit signed integers.
NC_FLOAT	32-bit IEEE floating-point.
NC_DOUBLE	64-bit IEEE floating-point.

These types are called "external", because they correspond to the portable external representation for NetCDF data. When a program reads external NetCDF data into an internal variable, the data is converted, if necessary, into the specified internal type. Similarly, if you write internal data into a NetCDF variable, this may cause it to be converted to a different external type, if the external type for the NetCDF variable differs from the internal type.

First versions of the R and C code of this package were based on the `netCDF` package by Thomas Lumley and the `ncdf` package by David Pierce. Milton Woods added some enhancements of the NetCDF library version 3.6.

A high-level interface based on this library is the `ncvar` package by Juerg Schmidli. It simplifies the handling of datasets which contain lots of metadata. Different metadata conventions are supported including the CF metadata conventions used by the climate modeling and forecasting community.

Note

The NetCDF and the UDUNITS library must be already installed on the system.

Author(s)

Pavel Michna

References

http://www.unidata.ucar.edu/software/netcdf/

http://www.unidata.ucar.edu/software/udunits/

sync.nc *Synchronize a NetCDF Dataset*

Description

Synchronize an open NetCDF dataset to disk.

Usage

```
sync.nc(ncfile)
```

Arguments

ncfile	Object of class "NetCDF" which points to the NetCDF dataset (as returned from open.nc).

Details

This function offers a way to synchronize the disk copy of a NetCDF dataset with in-memory buffers. There are two reasons one might want to synchronize after writes: To minimize data loss in case of abnormal termination, or to make data available to other processes for reading immediately after it is written.

Author(s)

Pavel Michna

References

http://www.unidata.ucar.edu/software/netcdf/

Examples

```
##  Create a new NetCDF dataset and define two dimensions
nc <- create.nc("foo.nc")

dim.def.nc(nc, "station", 5)
dim.def.nc(nc, "time", unlim=TRUE)

##  Create two variables, one as coordinate variable
var.def.nc(nc, "time", "NC_INT", "time")
var.def.nc(nc, "temperature", "NC_DOUBLE", c(0,1))

##  Define variable values
mytime        <- c(1:2)
mytemperature <- c(0.0, 1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8, 9.9)

##  Put the data
var.put.nc(nc, "time", mytime, 1, length(mytime))
```

```
var.put.nc(nc, "temperature", mytemperature, c(1,1), c(5,2))

##  Synchronize to disk
sync.nc(nc)

##  Now the data can be read
var.get.nc(nc, 0)
var.get.nc(nc, "temperature")

close.nc(nc)
```

`utcal.nc` *Convert Temporal Amounts to UTC Referenced Dates*

Description

Convert temporal amounts to UTC referenced date and time.

Usage

```
utcal.nc(unitstring, value, type="n")
```

Arguments

`unitstring`	A temporal unit with an origin (e.g., "days since 1900-01-01").
`value`	An amount (quantity) of the given temporal unit.
`type`	Character string which determines the output type. Can be `n` for numeric, `s` for string or `c` for POSIXct output.

Details

Converts the amount, `value`, of the temporal unit, `unitstring`, into a UTC-referenced date and time.

The UDUNITS package uses a mixed Gregorian/Julian calendar system. Dates prior to 1582-10-15 are assumed to use the Julian calendar, which was introduced by Julius Caesar in 46 BCE and is based on a year that is exactly 365.25 days long. Dates on and after 1582-10-15 are assumed to use the Gregorian calendar, which was introduced on that date and is based on a year that is exactly 365.2425 days long. (A year is actually approximately 365.242198781 days long.) Seemingly strange behavior of the UDUNITS package can result if a user-given time interval includes the changeover date.

Value

If the output type is set to numeric, result is a matrix containing the corresponding date(s) and time(s), with the following columns: year, month, day, hour, minute, second. If the output type is string, result is a vector of strings in the form `"YYYY-MM-DD hh:mm:ss"`. Otherwise result is a vector of POSIXct values.

Author(s)

Pavel Michna

References

http://www.unidata.ucar.edu/software/udunits/

Examples

```
##  Convert units to UTC referenced time
utcal.nc("hours since 1900-01-01 00:00:00 +01:00", c(0:5))
utcal.nc("hours since 1900-01-01 00:00:00 +01:00", c(0:5), type="s")
utcal.nc("hours since 1900-01-01 00:00:00 +01:00", c(0:5), type="c")
```

---

utinit.nc *Initialize the UDUNITS Library*

Description

Initialize the UDUNITS library.

Usage

```
utinit.nc(path="")
```

Arguments

path	Path to a units file containing initializing unit definitions.

Details

This function initializes the UDUNITS library. It is called by .First.lib when the package is loaded. Normally, the user does not need to call this function.

If path is non-NULL and not empty, then it specifies a units file containing initializing unit definitions; otherwise, the environment variable UDUNITS_PATH is checked and, if it exists and is not empty, then it is assumed to contain the pathname of the units file; otherwise, a compile-time default pathname is used.

Author(s)

Pavel Michna

References

http://www.unidata.ucar.edu/software/udunits/

Examples

```
##  NOTE: The user will normally never need to call this function
utinit.nc()
```

`utinvcal.nc` *Convert UTC Referenced Dates Into Temporal Amounts*

Description

Convert a UTC referenced date into a temporal amount.

Usage

```
utinvcal.nc(unitstring, value)
```

Arguments

`unitstring`	A temporal unit with an origin (e.g., "days since 1900-01-01").
`value`	Dates to convert as a numeric vector or array, or a vector of strings or POSIXct values.

Details

Converts a UTC-referenced date and time into the amount, `value`, of the temporal unit, `unitstring`.

The UDUNITS package uses a mixed Gregorian/Julian calendar system. Dates prior to 1582-10-15 are assumed to use the Julian calendar, which was introduced by Julius Caesar in 46 BCE and is based on a year that is exactly 365.25 days long. Dates on and after 1582-10-15 are assumed to use the Gregorian calendar, which was introduced on that date and is based on a year that is exactly 365.2425 days long. (A year is actually approximately 365.242198781 days long.) Seemingly strange behavior of the UDUNITS package can result if a user-given time interval includes the changeover date.

If the dates are given in string form, the structure must be exactly `"YYYY-MM-DD hh:mm:ss"`.

A vector of POSIXct values is also accepted as input. These are converted to the specified units by a linear transformation, without an intermediate separation into date components.

Value

A vector containing the amount(s) of the temporal unit(s) corresponding to the given date(s).

Author(s)

Pavel Michna

References

http://www.unidata.ucar.edu/software/udunits/

Examples

```
##  Convert UTC referenced time to other time units
utinvcal.nc("hours since 1900-01-01 00:00:00 +01:00", c(1900,1,1,5,25,0))
utinvcal.nc("hours since 1900-01-01 00:00:00 +01:00", "1900-01-01 05:25:00")
utinvcal.nc("hours since 1900-01-01 00:00:00 +01:00", ISOdatetime(1900,1,1,5,25,0,tz="UTC"))
```

---

var.def.nc *Define a NetCDF Variable*

Description

Define a new NetCDF variable.

Usage

```
var.def.nc(ncfile, varname, vartype, dimensions)
```

Arguments

ncfile	Object of class "NetCDF" which points to the NetCDF dataset (as returned from open.nc).
varname	Variable name. Must begin with an alphabetic character, followed by zero or more alphanumeric characters including the underscore ("_"). Case is significant.
vartype	One of the set of predefined NetCDF external data types. The valid NetCDF external data types are NC_BYTE, NC_CHAR, NC_SHORT, NC_INT, NC_FLOAT, and NC_DOUBLE.
dimensions	Vector of ndims dimension IDs or their names corresponding to the variable dimensions or NA if a scalar variable should be created. If the ID (or name) of the unlimited dimension is included, it must be last.

Details

This function creates a new NetCDF variable. A NetCDF variable has a name, a type, and a shape, which are specified when it is defined. A variable may also have values, which are established later in data mode.

Ordinarily, the name, type, and shape are fixed when the variable is first defined. The name may be changed, but the type and shape of a variable cannot be changed. However, a variable defined in terms of the unlimited dimension can grow without bound in that dimension. The fastetst varying dimension has to be first in dimensions, the slowest varying dimension last (this is the same way as an array is defined in R; i.e., opposite to the CDL conventions).

A NetCDF variable in an open NetCDF dataset is referred to by a small integer called a variable ID. Variable IDs are 0, 1, 2,..., in the order in which the variables were defined within a NetCDF dataset.

Attributes may be associated with a variable to specify such properties as units.

Author(s)

Pavel Michna

References

http://www.unidata.ucar.edu/software/netcdf/

Examples

```
##  Create a new NetCDF dataset and define two dimensions
nc <- create.nc("foo.nc")

dim.def.nc(nc, "station", 5)
dim.def.nc(nc, "time", unlim=TRUE)

##  Create two variables, one as coordinate variable
var.def.nc(nc, "time", "NC_INT", "time")
var.def.nc(nc, "temperature", "NC_DOUBLE", c(0,1))

close.nc(nc)
```

---

var.get.nc *Get a NetCDF Variable*

Description

Read the contents of a NetCDF variable.

Usage

```
var.get.nc(ncfile, variable, start=NA, count=NA,
        na.mode=0, collapse=TRUE, unpack=FALSE, rawchar=FALSE)
```

Arguments

ncfile	Object of class "NetCDF" which points to the NetCDF dataset (as returned from open.nc).
variable	ID or name of the variable.
start	A vector of indices indicating where to start reading the values (beginning at 1). The length of this vector must equal the number of dimensions the variable has. Order is leftmost varying fastest (as got from print.nc; opposite to the CDL conventions). If not specified (start=NA), reading starts at index 1.
count	A vector of integers indicating the count of values to read along each dimension. Order is leftmost varying fastest (as got from print.nc; opposite to the CDL conventions). The length of this vector must equal the number of dimensions the variable has. If not specified (count=NA), the entire variable or all values along the corresponding dimension(s) are read.

na.mode	Set the mode for handling missing values (NA) in numeric variables: 0=accept _FillValue or missing_value attribute, 1=accept only _FillValue attribute, 2=accept only missing_value attribute, 3=no missing value conversion.
collapse	TRUE if degenerated dimensions (length=1) should be omitted.
unpack	Packed variables are unpacked if unpack=TRUE and the attributes add_offset and scale_factor are defined. Default is FALSE.
rawchar	This option only relates to NetCDF variables of type NC_CHAR. When rawchar is FALSE (default), a NetCDF variable of type NC_CHAR is converted to a character array in R. The character values are from the fastest-varying dimension of the NetCDF variable, so that the R character array has one fewer dimensions than the NC_CHAR array. If rawchar is TRUE, the bytes of NC_CHAR data are read into an R raw array of the same shape.

Details

This function returns the value of a variable. Numeric variables are always returned in R double precision, no matter what precision they have in the on-disk dataset. NetCDF variables of type NC_CHAR are returned as R character or raw variables, as specified by argument rawchar.

Values of NA are supported in numeric variables. Values in the data file that match the variable's missing value attribute (as defined in na.mode) are automatically converted to NA before being returned to the user. If na.mode=0 and both attributes are defined, the value of _FillValue is used.

To reduce the storage space required by a NetCDF file, numeric variables are sometimes "packed" into types of lower precision. The original data can be recovered (approximately) by multiplication of the stored values by attribute scale_factor followed by addition of attribute add_offset. This unpacking operation is performed automatically for variables with attributes scale_factor and add_offset if argument unpack is set to TRUE. If unpack is FALSE, values are read from each variable without alteration.

Data in a NetCDF file is conceived as being a multi-dimensional array. The number and length of dimensions is determined when the variable is created. The start and count indices that this routine takes indicate where the reading starts along each dimension, and the count of values along each dimension to read.

The argument collapse allows to keep degenerated dimensions (if set to FALSE). As default, array dimensions with length=1 are omitted (e.g., an array with dimensions [2,1,3,4] in the NetCDF dataset is returned as [2,3,4]).

Awkwardness arises mainly from one thing: NetCDF data are written with the last dimension varying fastest, whereas R works opposite. Thus, the order of the dimensions according to the CDL conventions (e.g., time, latitude, longitude) is reversed in the R array (e.g., longitude, latitude, time).

Value

A multidimensional array with a data type that depends on the NetCDF variable. For NetCDF variables of type NC_CHAR, the R type is either character or raw, as specified by argument rawchar. All other NetCDF variables are returned to R as type numeric (double precision).

The dimension order in the R array is reversed relative to the order reported by NetCDF commands such as ncdump, because NetCDF arrays are stored in row-major (C) order whereas R arrays are stored in column-major (Fortran) order.

Arrays of type character drop the fastest-varying dimension of the corresponding NC_CHAR array, because this dimension corresponds to the length of the individual character elements. For example, an NC_CHAR array with dimensions (5,10) would be returned as a character vector containing 5 elements, each with a maximum length of 10 characters.

Note

NC_BYTE is always interpreted as signed.

Author(s)

Pavel Michna, Milton Woods

References

http://www.unidata.ucar.edu/software/netcdf/

Examples

```
##  Create a new NetCDF dataset and define two dimensions
nc <- create.nc("foo.nc")

dim.def.nc(nc, "station", 5)
dim.def.nc(nc, "time", unlim=TRUE)
dim.def.nc(nc, "max_string_length", 32)

##  Create three variables, one as coordinate variable
var.def.nc(nc, "time", "NC_INT", "time")
var.def.nc(nc, "temperature", "NC_DOUBLE", c(0,1))
var.def.nc(nc, "name", "NC_CHAR", c("max_string_length", "station"))

##  Put some missing_value attribute for temperature
att.put.nc(nc, "temperature", "missing_value", "NC_DOUBLE", -99999.9)

##  Define variable values
mytime        <- c(1:2)
mytemperature <- c(1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, NA, NA, 9.9)
myname        <- c("alfa", "bravo", "charlie", "delta", "echo")

##  Put the data
var.put.nc(nc, "time", mytime, 1, length(mytime))
var.put.nc(nc, "temperature", mytemperature, c(1,1), c(5,2))
var.put.nc(nc, "name", myname, c(1,1), c(32,5))

sync.nc(nc)

##  Get the data (or a subset)
var.get.nc(nc, 0)
var.get.nc(nc, "temperature")
var.get.nc(nc, "temperature", c(NA,2), c(NA,1))
var.get.nc(nc, "name")
var.get.nc(nc, "name", c(1,2), c(4,2))
```

```
var.get.nc(nc, "name", c(1,2), c(NA,2))

close.nc(nc)
```

`var.inq.nc` *Inquire About a NetCDF Variable*

Description

Inquire about a NetCDF variable.

Usage

```
var.inq.nc(ncfile, variable)
```

Arguments

`ncfile`	Object of class `"NetCDF"` which points to the NetCDF dataset (as returned from `open.nc`).
`variable`	Either the ID or the name of the variable to be inquired.

Details

This function returns information about a NetCDF variable. Information about a variable include its name, its ID, its type, its number of dimensions, a vector of the dimension IDs of this variable and the number of attributes. The valid external NetCDF data types are `NC_BYTE`, `NC_CHAR`, `NC_SHORT`, `NC_INT`, `NC_FLOAT`, and `NC_DOUBLE`.

Value

A list containing the following components:

`id`	Variable ID.
`name`	Variable name.
`type`	External NetCDF data type.
`ndims`	Number of dimensions the variable was defined as using.
`dimids`	Vector of dimension IDs corresponding to the variable dimensions (`NA` for scalar variables). Order is leftmost varying fastest.
`natts`	Number of variable attributes assigned to this variable.

Author(s)

Pavel Michna

References

http://www.unidata.ucar.edu/software/netcdf/

Examples

```
##  Create a new NetCDF dataset and define two dimensions
nc <- create.nc("foo.nc")

dim.def.nc(nc, "station", 5)
dim.def.nc(nc, "time", unlim=TRUE)

##  Create two variables, one as coordinate variable
var.def.nc(nc, "time", "NC_INT", "time")
var.def.nc(nc, "temperature", "NC_DOUBLE", c(0,1))

##  Inquire about these variables
var.inq.nc(nc, 0)
var.inq.nc(nc, "temperature")

close.nc(nc)
```

---

var.put.nc *Put Data Into a NetCDF Variable*

Description

Write the contents of a NetCDF variable.

Usage

```
var.put.nc(ncfile, variable, data, start=NA, count=NA, na.mode=0, pack=FALSE)
```

Arguments

ncfile	Object of class "NetCDF" which points to the NetCDF dataset (as returned from open.nc).
variable	ID or name of the variable.
data	The (multidimensional) array containing the data to write.
start	A vector of indices indicating where to start writing the passed values (beginning at 1). The length of this vector must equal the number of dimensions the variable has. Order is leftmost varying fastest (as got from print.nc; opposite to the CDL conventions). If set to NA, writing starts for each dimension at position 1.
count	A vector of integers indicating the count of values to write along each dimension. Order is leftmost varying fastest (as got from print.nc; opposite to the CDL conventions). The length of this vector must equal the number of dimensions the variable has. If set to NA, the dimesions are taken from data.
na.mode	Set the mode for handling missing values (NA) in numeric variables: 0=accept _FillValue or missing_value attribute, 1=accept only _FillValue attribute, 2=accept only missing_value attribute.
pack	Variables are packed if pack=TRUE and the attributes add_offset and scale_factor are defined. Default is FALSE.

Details

This function writes values to a NetCDF variable. Type conversion is performed by the NetCDF library, so that numeric values in R are automatically converted to the correct type of NetCDF variable.

However, text represented by R types `raw` and `character` can only be written to NetCDF type `NC_CHAR`. The dimensions of R `raw` variables map directly to NetCDF dimensions, but `character` variables have an implied dimension corresponding to the string length. This implied dimension must be defined explicitly as the fastest-varying dimension of the `NC_CHAR` variable, and it must be included as the first element of arguments `start` and `count` taken by this function.

Values of `NA` are supported in numeric variables if the variable's missing value attribute (as defined in `na.mode`) is set. They are converted to the corresponding value before writing to disk. If `na.mode=0` and both attributes are defined, the value of `_FillValue` is used.

To reduce the storage space required by a NetCDF file, numeric variables can be "packed" into types of lower precision. The packing operation involves subtraction of attribute `add_offset` before division by attribute `scale_factor`. This packing operation is performed automatically for variables defined with the two attributes `add_offset` and `scale_factor` if argument `pack` is set to `TRUE`. If `pack` is `FALSE`, `data` values are assumed to be packed correctly and are written to the variable without alteration.

Data in a NetCDF file is conceived as being a multi-dimensional array. The number and length of dimensions is determined when the variable is created. The `start` and `count` indices that this routine takes indicate where the writing starts along each dimension, and the count of values along each dimension to write.

Awkwardness arises mainly from one thing: NetCDF data are written with the last dimension varying fastest, whereas R works opposite. Thus, the order of the dimensions according to the CDL conventions (e.g., time, latitude, longitude) is reversed in the R array (e.g., longitude, latitude, time).

Note

`NC_BYTE` is always interpreted as signed. For best performance, it is recommended that the definition of dimensions, variables and attributes is completed before variables are read or written.

Author(s)

Pavel Michna, Milton Woods

References

http://www.unidata.ucar.edu/software/netcdf/

Examples

```
##  Create a new NetCDF dataset and define two dimensions
nc <- create.nc("foo.nc")

dim.def.nc(nc, "station", 5)
dim.def.nc(nc, "time", unlim=TRUE)
dim.def.nc(nc, "max_string_length", 32)
```

```
##  Create three variables, one as coordinate variable
var.def.nc(nc, "time", "NC_INT", "time")
var.def.nc(nc, "temperature", "NC_DOUBLE", c(0,1))
var.def.nc(nc, "name", "NC_CHAR", c("max_string_length", "station"))

##  Put some missing_value attribute for temperature
att.put.nc(nc, "temperature", "missing_value", "NC_DOUBLE", -99999.9)

##  Define variable values
mytime        <- c(1:2)
mytemperature <- c(1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, NA, NA, 9.9)
myname        <- c("alfa", "bravo", "charlie", "delta", "echo")

dim(mytemperature) <- c(5,2)

##  Put the data with indicated start/count
var.put.nc(nc, "time", mytime, 1, length(mytime))
var.put.nc(nc, "temperature", mytemperature, c(1,1), c(5,2))
var.put.nc(nc, "name", myname, c(1,1), c(32,5))

sync.nc(nc)

##  Put the data with default start/count
var.put.nc(nc, "time", mytime)
var.put.nc(nc, "temperature", mytemperature)
var.put.nc(nc, "name", myname)

close.nc(nc)
```

`var.rename.nc` *Rename a NetCDF Variable*

Description

Rename a NetCDF variable.

Usage

```
var.rename.nc(ncfile, variable, newname)
```

Arguments

`ncfile`	Object of class `"NetCDF"` which points to the NetCDF dataset (as returned from `open.nc`).
`variable`	Either the ID or the name of the variable to be renamed.
`newname`	The new variable name.

Details

This function renames an existing variable in a NetCDF dataset open for writing. A variable cannot be renamed to have the same name as another variable.

Author(s)

Pavel Michna

References

http://www.unidata.ucar.edu/software/netcdf/

Examples

```
##  Create a new NetCDF dataset and define two dimensions
nc <- create.nc("foo.nc")

dim.def.nc(nc, "station", 5)
dim.def.nc(nc, "time", unlim=TRUE)

##  Create two variables, one as coordinate variable
var.def.nc(nc, "time", "NC_INT", "time")
var.def.nc(nc, "temperature", "NC_DOUBLE", c(0,1))

##  Rename these variables
var.rename.nc(nc, 0, "mytime")
var.rename.nc(nc, "temperature", "mytemperature")

close.nc(nc)
```

Index