Package: RNetCDF (via r-universe)

August 27, 2024

Version 2.9-2

Date 2024-03-24

Title Interface to 'NetCDF' Datasets

Depends R (>= 3.0.0)

SystemRequirements netcdf (>=4.1.3), udunits-2 (>=2.0.4)

Suggests bit64, tools

Enhances pbdMPI, Rmpi

Description An interface to the 'NetCDF' file formats designed by Unidata for efficient storage of array-oriented scientific data and descriptions. Most capabilities of 'NetCDF' version 4 are supported. Optional conversions of time units are enabled by 'UDUNITS' version 2, also from Unidata.

License GPL (>= 2) | file LICENSE

URL https://github.com/mjwoods/RNetCDF
https://www.unidata.ucar.edu/software/netcdf/
https://www.unidata.ucar.edu/software/udunits/

BugReports https://github.com/mjwoods/RNetCDF/issues

NeedsCompilation yes

Repository https://mjwoods.r-universe.dev

RemoteUrl https://github.com/mjwoods/rnetcdf

RemoteRef HEAD

RemoteSha 3b1cdc9cf722705c1338fab555b8033469cacf80

Contents

`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.

Valid attribute ID numbers range from 0 to the number of attributes minus 1. The number of attributes of a file, group, or variable can be found using the relevant inquiry function (file.inq.nc, grp.inq.nc, or var.inq.nc).

Author(s)

Pavel Michna, Milton Woods

References

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

Examples

```
##  Create two new NetCDF datasets and define two dimensions
file1 <- tempfile("att.copy_", fileext=".nc")
file2 <- tempfile("att.copy_", fileext=".nc")
nc.1 <- create.nc(file1)
nc.2 <- create.nc(file2)

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", "_FillValue", "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)
unlink(file1)
unlink(file2)
```

`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 or ID of the attribute to be deleted.

Details

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

Valid attribute ID numbers range from 0 to the number of attributes minus 1. The number of attributes of a file, group, or variable can be found using the relevant inquiry function (`file.inq.nc`, `grp.inq.nc`, or `var.inq.nc`).

Author(s)

Pavel Michna, Milton Woods

References

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

Examples

```
##  Create a new NetCDF dataset and define two dimensions
file1 <- tempfile("att.delete_", fileext=".nc")
nc <- create.nc(file1)

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", "_FillValue", "NC_DOUBLE", -99999.9)
att.put.nc(nc, "NC_GLOBAL", "title", "NC_CHAR", "Data from Foo")

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

close.nc(nc)
unlink(file1)
```

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

Description

Get an attribute from a NetCDF dataset.

Usage

```
att.get.nc(ncfile, variable, attribute, rawchar=FALSE, fitnum=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 from which the attribute will be read, or `"NC_GLOBAL"` for a global attribute.
`attribute`	Attribute name or ID.
`rawchar`	This option only relates to NetCDF attributes of type `NC_CHAR`. When `rawchar` is `FALSE` (default), a NetCDF attribute of type `NC_CHAR` is converted to a `character` string in R. If `rawchar` is `TRUE`, the bytes of `NC_CHAR` data are read into an R `raw` vector.

`fitnum` By default, all numeric variables are read into R as double precision values. When `fitnum==TRUE`, the smallest R numeric type that can exactly represent each external type is used, as follows:

NC_BYTE	integer
NC_UBYTE	integer
NC_SHORT	integer
NC_USHORT	integer
NC_INT	integer
NC_UINT	double
NC_FLOAT	double
NC_DOUBLE	double
NC_INT64	integer64
NC_UINT64	integer64

Details

This function returns the value of the attribute.

Value

Vector 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`. For `NC_STRING`, the R type is `character`. Numeric variables are read as double precision by default, but the smallest R type that exactly represents each external type is used if `fitnum` is `TRUE`.

Valid attribute ID numbers range from 0 to the number of attributes minus 1. The number of attributes of a file, group, or variable can be found using the relevant inquiry function (`file.inq.nc`, `grp.inq.nc`, or `var.inq.nc`).

Note

`NC_BYTE` is always interpreted as signed.

Author(s)

Pavel Michna, Milton Woods

References

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

Examples

```
##  Create a new NetCDF dataset and define two dimensions
file1 <- tempfile("att.get_", fileext=".nc")
nc <- create.nc(file1)

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", "_FillValue", "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", "_FillValue")
att.get.nc(nc, "temperature", "long_name")
att.get.nc(nc, "NC_GLOBAL", "title")
att.get.nc(nc, "NC_GLOBAL", "history")

close.nc(nc)
unlink(file1)
```

`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. In general, attributes are accessed by name rather than by their ID number because the attribute number is more volatile than the name, since it can change when other attributes of the same variable are deleted.

Valid attribute ID numbers range from 0 to the number of attributes minus 1. The number of attributes of a file, group, or variable can be found using the relevant inquiry function (`file.inq.nc`, `grp.inq.nc`, or `var.inq.nc`).

Value

A list containing the following components:

id	Attribute ID.
name	Attribute name.
type	External NetCDF data type as one of the following labels: NC_BYTE, NC_UBYTE, NC_CHAR, NC_SHORT, NC_USHORT, NC_INT, NC_UINT, NC_INT64, NC_UINT64, NC_FLOAT, NC_DOUBLE, NC_STRING, or a user-defined type name.
length	Length of this attribute.

Author(s)

Pavel Michna, Milton Woods

References

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

Examples

```
##  Create a new NetCDF dataset and define two dimensions
file1 <- tempfile("att.inq_", fileext=".nc")
nc <- create.nc(file1)

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", "_FillValue", "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", "_FillValue")
att.inq.nc(nc, "NC_GLOBAL", "title")

close.nc(nc)
unlink(file1)
```

---

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	External NetCDF data type as one of the following labels: NC_BYTE, NC_UBYTE, NC_CHAR, NC_SHORT, NC_USHORT, NC_INT, NC_UINT, NC_INT64, NC_UINT64, NC_FLOAT, NC_DOUBLE, NC_STRING, or a user-defined type name.
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.

Text represented by R type character can be written to NetCDF types NC_CHAR and NC_STRING, and R type raw can be written to NetCDF type NC_CHAR.

R numeric and integer variables can be written to NetCDF numeric types. The NetCDF library handles type conversions, but conversions of values outside the range of a type will result in an error. Due to the lack of native support for 64-bit integers in R, this function accepts integer64 vectors.

Note

NC_BYTE is always interpreted as signed.

Author(s)

Pavel Michna, Milton Woods

References

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

Examples

```
##  Create a new NetCDF dataset and define two dimensions
file1 <- tempfile("att.put_", fileext=".nc")
nc <- create.nc(file1)

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", "_FillValue", "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)
unlink(file1)
```

---

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.

Valid attribute ID numbers range from 0 to the number of attributes minus 1. The number of attributes of a file, group, or variable can be found using the relevant inquiry function (file.inq.nc, grp.inq.nc, or var.inq.nc).

Author(s)

Pavel Michna, Milton Woods

References

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

Examples

```
##  Create a new NetCDF dataset and define two dimensions
file1 <- tempfile("att.rename_", fileext=".nc")
nc <- create.nc(file1)

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", "_FillValue", "NC_DOUBLE", -99999.9)
att.put.nc(nc, "NC_GLOBAL", "title", "NC_CHAR", "Data from Foo")

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

close.nc(nc)
unlink(file1)
```

---

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, Milton Woods

References

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

Examples

```
##  Create a void NetCDF dataset
file1 <- tempfile("close_", fileext=".nc")
nc <- create.nc(file1)
close.nc(nc)
unlink(file1)
```

---

config.nc *Configured options*

Description

Find NetCDF options detected when installing RNetCDF.

Usage

```
config.nc()
```

Details

This function is not intended for user code, and it is subject to change or removal without notice. It is currently needed for RNetCDF package tests, to determine expected behaviour of the NetCDF C library.

Unless otherwise documented, optional NetCDF features that are not detected when installing RNetCDF will raise an error when called from R code. If necessary, work arounds can be implemented by wrapping the relevant code in `try` or `tryCatch`.

Author(s)

Pavel Michna, Milton Woods

`create.nc` *Create a NetCDF Dataset*

Description

Create a new NetCDF dataset.

Usage

```
create.nc(filename, clobber=TRUE, share=FALSE, prefill=TRUE,
        format="classic", large=FALSE, diskless=FALSE, persist=FALSE,
        mpi_comm=NULL, mpi_info=NULL)
```

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`.
`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 `TRUE`.
`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`.
`format`	The file format. One of `"classic"`, `"offset64"`, `"data64"`, `"netcdf4"` or `"classic4"`. See below for details.
`large`	(Deprecated) `large=TRUE` sets the file format to `"offset64"` when `format="classic"`.
`diskless`	When `diskless=TRUE`, the file is created in memory without writing to disk. This allows netcdf datasets to be used as fast, temporary files. When the file is closed, the contents are lost unless `persist=TRUE`.
`persist`	When `persist=TRUE`, a file created with `diskless=TRUE` is flushed to disk when closed. In some cases, this may be faster than manipulating files directly on disk.
`mpi_comm`	Fortran handle of MPI communicator for parallel I/O. The default of `NULL` implies serial I/O. Valid Fortran handles may be obtained from your chosen MPI package for R - for example comm.c2f or mpi.comm.c2f.
`mpi_info`	Fortran handle of MPI Info object for parallel I/O. The default value `NULL` specifies that the `MPI_INFO_NULL` object is used. Other valid Fortran handles may be obtained from your chosen MPI package for R - for example info.c2f.

Details

This function creates a new NetCDF dataset, returning an object of class `NetCDF` that can be used in R.

The file format is specified by the `format` argument, which may take the following values:

`"classic"` (default) Original netcdf file format, still widely used and recommended for maximum portability of datasets. Uses a signed 32-bit offset in its internal structures, so files larger than 2GB can only be created under limited conditions.

`"offset64"` 64-bit offset extension of original format, introduced by netcdf-3.6. Allows larger files and variables than `"classic"` format, but there remain some restrictions on files larger than 2GB.

`"data64"` Extension of `"classic"` format to support large files (i.e. over 2GB) and large variables (over 2B array elements). This format was introduced in netcdf-4.4.0.

`"netcdf4"` Netcdf in an HDF5 container, introduced by netcdf-4.0. Allows dataset sizes up to filesystem limits, and extends the feature set of the older formats.

`"classic4"` Same file format as `"netcdf4"`, but this option ensures that only classic netcdf data structures are stored in the file for compatibility with older software (when linked with the netcdf4 library).

Value

Object of class `NetCDF` which points to the NetCDF dataset, returned invisibly.

Author(s)

Pavel Michna, Milton Woods

References

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

Examples

```
##  Create empty NetCDF datasets with different formats
file1 <- tempfile("create3_", fileext=".nc")
nc <- create.nc(file1)
close.nc(nc)
unlink(file1)

file2 <- tempfile("create64_", fileext=".nc")
nc2 <- create.nc(file2,format="offset64")
close.nc(nc2)
unlink(file2)

file3 <- tempfile("create4_", fileext=".nc")
nc3 <- create.nc(file3,format="netcdf4")
close.nc(nc3)
unlink(file3)
```

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.

Value

NetCDF variable identifier, returned invisibly.

Author(s)

Pavel Michna, Milton Woods

References

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

Examples

```
##  Create a new NetCDF dataset and define two dimensions
file1 <- tempfile("dim.def_", fileext=".nc")
nc <- create.nc(file1)

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

close.nc(nc)
unlink(file1)
```

`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, Milton Woods

References

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

Examples

```
##  Create a new NetCDF dataset and define two dimensions
file1 <- tempfile("dim.inq_", fileext=".nc")
nc <- create.nc(file1)

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)
unlink(file1)
```

---

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, Milton Woods

References

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

Examples

```
##  Create a new NetCDF dataset and define two dimensions
file1 <- tempfile("dim.rename_", fileext=".nc")
nc <- create.nc(file1)

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)
unlink(file1)
```

`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`).

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.
`format`	Format of file, typically `"classic"`, `"offset64"`, `"data64"`, `"classic4"` or `"netcdf4"`.
`libvers`	Version string of the NetCDF library in the current R session.

Author(s)

Pavel Michna, Milton Woods

References

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

Examples

```
##  Create a new NetCDF dataset and define two dimensions
file1 <- tempfile("file.inq_", fileext=".nc")
nc <- create.nc(file1)

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", "_FillValue", "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)
unlink(file1)
```

---

grp.def.nc *Define a NetCDF Group*

Description

Define a NetCDF Group.

Usage

```
grp.def.nc(ncid, grpname)
```

Arguments

ncid	Object of class NetCDF which points to the NetCDF dataset (as returned from open.nc) or parent group (as returned by this function).
grpname	Group name. Must begin with an alphabetic character, followed by zero or more alphanumeric characters including the underscore ("_"). Case is significant.

Details

This function may only be used with datasets in `netcdf4` format. It creates a new NetCDF group, which may be used as a container for other NetCDF objects, including groups, dimensions, variables and attributes.

Most NetCDF object types, including groups, variables and global attributes, are visible only in the group where they are defined. However, dimensions are visible in their groups and all child groups.

Value

Object of class `NetCDF` which points to the NetCDF group, returned invisibly.

Author(s)

Pavel Michna, Milton Woods

References

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

Examples

```
##  Create a new NetCDF4 dataset
file1 <- tempfile("grp.def_", fileext=".nc")
nc <- create.nc(file1, format="netcdf4")

## Define dimensions, variables and attributes in the root group
dim.def.nc(nc, "station", 5)
var.def.nc(nc, "station", "NC_CHAR", c("station"))
att.put.nc(nc, "NC_GLOBAL", "Description", "NC_CHAR", "Site-based measurements")

## Define a group
grp <- grp.def.nc(nc, "time_series")

## Define dimensions and variables in the new group
dim.def.nc(grp, "time", unlim=TRUE)
var.def.nc(grp, "time", "NC_INT", "time")
var.def.nc(grp, "temperature", "NC_DOUBLE", c("station","time"))
att.put.nc(nc, "NC_GLOBAL", "Description", "NC_CHAR", "Time-series at sites")

close.nc(nc)
unlink(file1)
```

`grp.inq.nc` *Inquire About a NetCDF Group*

Description

Inquire about a NetCDF group.

Usage

```
grp.inq.nc(ncid,grpname=NULL,ancestors=TRUE)
```

Arguments

ncid	Object of class NetCDF which points to a NetCDF group (from grp.def.nc) or dataset (from open.nc).
grpname	By default, the inquiry relates to the group represented by ncid. If grpname is a character string, a group with this name is examined instead. A hierarchical search is performed if grpname contains "/", otherwise only the immediate group of ncid is searched for a matching group name.
ancestors	If TRUE, dimensions and names of ancestor groups are examined. Otherwise, only dimensions and names defined in the current group are reported.

Details

This function provides information about the structure of a NetCDF group or dataset. The results allow programs to explore a dataset without prior knowledge of the contents.

Value

A list containing the following components:

self	Object of class NetCDF representing the group.
parent	Object of class NetCDF representing the parent group, or NULL for the root group.
grps	List of objects of class NetCDF representing the groups in the group.
name	Name of the NetCDF group.
fullname	Full name of the NetCDF group, with ancestors listed in order from the root group of the dataset and separated by "/". Omitted if ancestors is FALSE.
dimids	Vector of dimension identifiers. If ancestors is TRUE (default), all visible dimensions in the group and its ancestors are reported, otherwise only dimensions defined in the group of ncid are shown.
unlimids	Vector of identifiers for unlimited dimensions. If ancestors is TRUE (default), all unlimited dimensions in the group and its ancestors are reported, otherwise only unlimited dimensions defined in the group of ncid are shown.
varids	Vector of identifiers for variables in the group.
typeids	Vector of identifiers for types in the group.
ngatts	Number of group attributes.

Author(s)

Pavel Michna, Milton Woods

References

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

Examples

```
# Create a new NetCDF dataset:
file1 <- tempfile("grp.inq_", fileext=".nc")
nc <- create.nc(file1, format="netcdf4")

# Define groups in root group.
# (Any names can be used; hierarchical numbers are used here for clarity)
grp11 <- grp.def.nc(nc, "group1.1")
grp12 <- grp.def.nc(nc, "group1.2")

# Define group nested in group1.1:
grp111 <- grp.def.nc(grp11, "group1.1.1")

# Put some attributes in each group.
# (We could also define dimensions, types, and variables).
att.put.nc(nc, "NC_GLOBAL", "title", "NC_CHAR", "Group 1 (root)")
att.put.nc(grp11, "NC_GLOBAL", "title", "NC_CHAR", "Group 1.1")
att.put.nc(grp12, "NC_GLOBAL", "title", "NC_CHAR", "Group 1.2")
att.put.nc(grp111, "NC_GLOBAL", "title", "NC_CHAR", "Group 1.1.1")

## Examine contents of a group directly using its hierarchical name ...

mygrp <- grp.inq.nc(nc, "/group1.1/group1.1.1")
att.get.nc(mygrp$self, "NC_GLOBAL", "title")

## Recursively examine contents of nested groups ...
# (See also print.nc for a visual overview)

get_global_atts <- function(ncid) {
  inq <- grp.inq.nc(ncid)
  atts <- character(inq$ngatts)
  for (ii in seq_len(inq$ngatts)) {
    atts[ii] <- att.get.nc(ncid, "NC_GLOBAL", ii-1)
  }
  ngrps <- length(inq$grps)
  grps <- vector("list", ngrps + 1)
  grps[[1]] <- atts
  for (ii in seq_len(ngrps)) {
    grps[[ii + 1]] <- get_global_atts(inq$grps[[ii]])
  }
  return(grps)
}

get_global_atts(nc)

## Tidy up:
close.nc(nc)
unlink(file1)
```

`grp.rename.nc` *Rename a NetCDF Group*

Description

Rename a NetCDF group.

Usage

```
grp.rename.nc(ncid, newname, oldname=NULL)
```

Arguments

ncid	Object of class NetCDF which points to a NetCDF group (from grp.def.nc) or dataset (from open.nc).
newname	The new group name.
oldname	By default, the rename applies to the group represented by ncid. If oldname is a character string, a group with this name is renamed instead. A hierarchical search is performed if oldname contains "/", otherwise only the immediate group of ncid is searched for a matching group name.

Details

This function renames an existing group in a dataset of "netcdf4" format that is open for writing. A group cannot be renamed to have the same name as another group, type or variable in the parent group.

Author(s)

Pavel Michna, Milton Woods

References

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

Examples

```
##  Create a new NetCDF dataset and define a group
file1 <- tempfile("grp.rename_", fileext=".nc")
nc <- create.nc(file1, format="netcdf4")

grp <- grp.def.nc(nc, "oldgroup")

##  Rename the group (operation not supported by early versions of the netcdf4 library)
try(grp.rename.nc(grp, "newgroup"))

close.nc(nc)
unlink(file1)
```

`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, diskless=FALSE, persist=FALSE,
        mpi_comm=NULL, mpi_info=NULL, ...)
```

Arguments

`con`	Filename of the NetCDF dataset to be opened. If the underlying NetCDF library supports OPeNDAP, `con` may be an OPeNDAP URL.
`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 `TRUE`.
`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`.
`diskless`	When `diskless=TRUE`, the file is read entirely into memory, and any changes are kept in memory without writing to disk. The netcdf library may ignore this option for files in netcdf4 format.
`persist`	When `persist=TRUE`, a file opened with `diskless=TRUE` is flushed to disk when closed. In some cases, this may be faster than manipulating files directly on disk.
`mpi_comm`	Fortran handle of MPI communicator for parallel I/O. The default of `NULL` implies serial I/O. Valid Fortran handles may be obtained from your chosen MPI package for R - for example comm.c2f or mpi.comm.c2f.
`mpi_info`	Fortran handle of MPI Info object for parallel I/O. The default value `NULL` specifies that the `MPI_INFO_NULL` object is used. Other valid Fortran handles may be obtained from your chosen MPI package for R - for example info.c2f.
`...`	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, returned invisibly.

Author(s)

Pavel Michna, Milton Woods

References

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

Examples

```
##  Create a void NetCDF dataset
file1 <- tempfile("open_", fileext=".nc")
nc <- create.nc(file1)
close.nc(nc)

##  Open the NetCDF dataset for writing
nc <- open.nc(file1, write=TRUE)
close.nc(nc)
unlink(file1)
```

---

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 the structure of a NetCDF dataset, including lists of all groups, dimensions, user-defined types, variables and attributes.

The output of this function is similar to the ncdump -h command supplied with the NetCDF C library. One important difference is that array dimensions are shown by print.nc in the order used by R, where the leftmost subscript varies fastest.

Author(s)

Pavel Michna, Milton Woods

References

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

Examples

```
##  Create a new NetCDF dataset
file1 <- tempfile("print_", fileext=".nc")
nc <- create.nc(file1, format="netcdf4")

## Create a group (just because we can!):
grp <- grp.def.nc(nc, "data")

## Create some dimensions, putting one inside the group:
dim.def.nc(nc, "time", unlim=TRUE)
dim.def.nc(grp, "station", 5)

##  Create two variables, putting one inside the group:
var.def.nc(nc, "time", "NC_INT", "time")
var.def.nc(grp, "temperature", "NC_DOUBLE", c("station","time"))

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

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

close.nc(nc)
unlink(file1)
```

---

read.nc *Read a NetCDF Dataset*

Description

Read all data from a NetCDF dataset.

Usage

```
read.nc(ncfile, recursive=FALSE, ...)
```

Arguments

ncfile	Object of class NetCDF which points to the NetCDF dataset (as returned from open.nc).
recursive	Descend recursively into any groups in the dataset if TRUE.
...	Optional arguments passed to var.get.nc.

Details

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

Groups in the dataset may optionally be read recursively and returned as nested lists. Each list has the name of the corresponding group in the dataset.

Value

A list with the list elements containing an array for each variable or a (possibly nested) list for each group in the NetCDF dataset.

Author(s)

Pavel Michna, Milton Woods

References

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

Examples

```
##  Create a new NetCDF dataset
file1 <- tempfile("read_", fileext=".nc")
nc <- create.nc(file1, format="netcdf4")

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

##  Create two coordinate variables
var.def.nc(nc, "time", "NC_INT", "time")
var.def.nc(nc, "name", "NC_CHAR", c("max_string_length", "station"))

## Create a group to contain the data
# This is not necessary, but shows it can be done.
grp <- grp.def.nc(nc, "data")

## Create a data variable
var.def.nc(grp, "temperature", "NC_DOUBLE", c("station","time"))

##  Put some _FillValue attribute for temperature
att.put.nc(grp, "temperature", "_FillValue", "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, "name", myname, c(1,1), c(32,5))
var.put.nc(grp, "temperature", mytemperature, c(1,1), c(5,2))

sync.nc(nc)

##  Read the dataset, including the contents of any groups
read.nc(nc, recursive=TRUE)

close.nc(nc)
unlink(file1)
```

RNetCDF *R Interface to NetCDF Datasets*

Description

This package provides an interface to Unidata's NetCDF library functions (version 4) and furthermore access to Unidata's UDUNITS (version 2) 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.

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 versions 3.6 and 4.x.

Functions

Help pages are available for the following RNetCDF functions:

Category	Function
Dataset	close.nc
	create.nc
	file.inq.nc
	open.nc
	print.nc
	read.nc

	sync.nc
Group	grp.def.nc
	grp.inq.nc
	grp.rename.nc
Attribute	att.copy.nc
	att.delete.nc
	att.get.nc
	att.inq.nc
	att.put.nc
	att.rename.nc
Dimension	dim.def.nc
	dim.inq.nc
	dim.rename.nc
Data type	type.def.nc
	type.inq.nc
Variable	var.def.nc
	var.get.nc
	var.inq.nc
	var.par.nc
	var.put.nc
	var.rename.nc
Calendar	utcal.nc
	utinit.nc
	utinvcal.nc

Data Types

The external types supported by all NetCDF datasets are:

NC_CHAR	8-bit characters intended for representing text.
NC_BYTE	8-bit signed 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.

Datasets in NetCDF4 format support additional external types, including:

NC_UBYTE	8-bit unsigned integers.
NC_USHORT	16-bit unsigned integers.
NC_UINT	32-bit unsigned integers.
NC_INT64	64-bit signed integers.
NC_UINT64	64-bit unsigned integers.
NC_STRING	variable length character strings.

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.

In addition to the external types, NetCDF4 supports user-defined types. See `type.def.nc` for more explanation.

Note

When installing RNetCDF from source code, the netcdf4 library and header files must be installed on the system. Calendar functions will only be enabled in RNetCDF if the udunits2 library and header files are detected during the build process. Parallel file access requires a netcdf4 library built with MPI support along with an MPI interface package installed in R (e.g. pbdMPI or Rmpi).

Author(s)

Pavel Michna, Milton Woods

References

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

https://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, Milton Woods

References

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

Examples

```
##  Create a new NetCDF dataset and define two dimensions
file1 <- tempfile("sync_", fileext=".nc")
nc <- create.nc(file1)

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)
dim(mytime) <- c(2)
mytemp <- c(0.0, 1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8, 9.9)
dim(mytemp) <- c(5,2)

##  Put the data
var.put.nc(nc, "time", mytime)
var.put.nc(nc, "temperature", mytemp)

##  Synchronize to disk
sync.nc(nc)

##  Open a new connection to the dataset and read data:
nc2 <- open.nc(file1)
newtime <- var.get.nc(nc2, 0)
newtemp <- var.get.nc(nc2, "temperature")
stopifnot(all.equal(newtime,mytime))
stopifnot(all.equal(newtemp,mytemp))

close.nc(nc)
close.nc(nc2)
unlink(file1)
```

---

type.def.nc *Define a NetCDF Type*

Description

Define complex data structures based on existing NetCDF data types.

Usage

```
type.def.nc(ncfile, typename, class, size=NULL, basetype=NULL,
            names=NULL, values=NULL, subtypes=NULL, dimsizes=NULL)
```

Arguments

ncfile	Object of class NetCDF which points to the NetCDF dataset (as returned from open.nc).
typename	Name to identify the new data type. Must begin with an alphabetic character, followed by zero or more alphanumeric characters including the underscore ("_"). Case is significant.
class	One of the keywords "compound", "enum", "opaque" or "vlen".
size	("opaque") Size in bytes of a single item of the opaque type.
basetype	("enum" or "vlen") Base type, given as the name or identifier of an existing NetCDF type. Only built-in integer types (e.g. "NC_INT") are allowed for class="enum".
names	("compound" or "enum") Name of each field or member (character vector).
values	("enum") Numeric value of each member (numeric vector).
subtypes	("compound") NetCDF type of each field, given by name (character vector) or identifier (numeric vector).
dimsizes	("compound") Array dimensions of each field, specified as a list of numeric vectors. Dimensions are given in R order (leftmost index varies fastest; opposite to CDL conventions). If a list item is NULL, the corresponding field is a scalar.

Details

User-defined types are supported by files in "netcdf4" format. This function creates a new NetCDF data type, which can be used in definitions of NetCDF variables and attributes.

Several varieties of data type are supported, as specified by argument class:

"compound"	Combines atomic and user-defined types into C-like structs.
"enum"	Set of named integer values, similar to an R factor.
"opaque"	Blobs of arbitrary data with a given size.
"vlen"	Variable length vectors of a given base type.

type.def.nc may be repeated to insert additional members of an "enum" type or fields of a "compound" type. However, the size of a "compound" type is calculated from the fields specified when it is first defined, and later insertion of fields will only succeed if there is sufficient free space after the last field. Existing fields/members cannot be modified, and types cannot be removed from a dataset.

Value

NetCDF type identifier, returned invisibly.

Author(s)

Pavel Michna, Milton Woods

References

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

Examples

```
##  Create a new NetCDF4 dataset and define types
file1 <- tempfile("type.def_", fileext=".nc")
nc <- create.nc(file1, format="netcdf4")

# Compound type:
type.def.nc(nc, "astruct", "compound",
            names=c("siteid", "height", "colour"),
            subtypes=c("NC_INT", "NC_DOUBLE", "NC_SHORT"),
            dimsizes=list(NULL, NULL, c(3)))

# Enum type:
type.def.nc(nc, "afactor", "enum", basetype="NC_INT",
            names=c("peanut butter", "jelly"),
            values=c(101,102))

# Opaque type:
type.def.nc(nc, "ablob", "opaque", size=128)

# Vlen type:
type.def.nc(nc, "avector", "vlen", basetype="NC_FLOAT")

close.nc(nc)
unlink(file1)
```

---

type.inq.nc *Inquire About a NetCDF Type*

Description

Inquire about a NetCDF builtin or user-defined data type.

Usage

```
type.inq.nc(ncfile, type, fields=TRUE)
```

Arguments

ncfile	Object of class NetCDF which points to the NetCDF dataset or group.
type	ID or name of a NetCDF data type.
fields	Read members of enum types or fields of compound types (default TRUE).

Details

This function obtains information about a NetCDF data type, which could be builtin or user-defined. The items in the return list depend on the class of the NetCDF type.

Value

A list containing the following components:

`id`	Type ID.
`name`	Type name.
`class`	One of the keywords `"builtin"`, `"compound"`, `"enum"`, `"opaque"` or `"vlen"`.
`size`	Size in bytes of a single item of the type (or a single element of a `"vlen"`).
`basetype`	(`"enum"` or `"vlen"`) Name of the NetCDF type of each element.

If `fields=TRUE`, the return list includes details about members of enum types or fields of compound types:

`value`	(`"enum"` only) Named vector with numeric values of all members.
`offset`	(`"compound"` only) Named vector with the offset of each field in bytes from the beginning of the compound type.
`subtype`	(`"compound"` only) Named vector with the NetCDF type name of each field.
`dimsizes`	(`"compound"` only) Named list with array dimensions of each field. Dimension lengths are reported in R order (leftmost index varies fastest; opposite to CDL conventions). A `NULL` length indicates a scalar.

Author(s)

Pavel Michna, Milton Woods

References

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

See Also

`grp.inq.nc` - get a list of NetCDF types defined in a dataset or group.

`type.def.nc` - define a new NetCDF type.

Examples

```
##  Create a new NetCDF4 dataset and define types
file1 <- tempfile("type.inq_", fileext=".nc")
nc <- create.nc(file1, format="netcdf4")

# Define a type of each class:
type.def.nc(nc, "blob", "opaque", size=128)
type.def.nc(nc, "vector", "vlen", basetype="NC_FLOAT")
```

```
type.def.nc(nc, "factor", "enum", basetype="NC_INT",
            names=c("peanut butter", "jelly"),
            values=c(101, 102))

type.def.nc(nc, "struct", "compound",
            names=c("siteid", "height", "colour"),
            subtypes=c("NC_INT", "NC_DOUBLE", "NC_SHORT"),
            dimsizes=list(NULL, NULL, c(3)))

# Inquire about the types:
typeids <- grp.inq.nc(nc)$typeids

for (typeid in typeids) {
  print(type.inq.nc(nc, typeid))
}

close.nc(nc)
unlink(file1)
```

`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.

Functions `utcal.nc` and `utinvcal.nc` provide a convenient way to convert time values between the forms used by NetCDF variables and R functions. Most R functions require times to be expressed as seconds since the beginning of 1970 in the UTC time zone, typically using objects of class `POSIXct` or `POSIXlt`. NetCDF files store times in numeric variables with a wide variety of units. The units and calendar are stored in attributes of the time variable, as described by the CF Conventions. Units are expressed as a string, in the form of a time unit since a fixed date-time (e.g. `"hours since 2000-01-01 00:00:00 +00:00"`, or more simply `"hours since 2000-01-01"`).

The conversions of times between units are performed by the UDUNITS library using 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.

Conversions involving alternative calendars are not supported by UDUNITS, but they can be performed by other R packages. For example, https://CRAN.R-project.org/package=PCICt implements 360- and 365-day calendars.

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, Milton Woods

References

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

http://cfconventions.org

See Also

utinvcal.nc

Examples

```
if (inherits(try(utcal.nc("seconds since 1970-01-01", 0)), "try-error")) {
  warning("UDUNITS calendar conversions not supported by this build of RNetCDF")

} else {

  ##  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")

  ## Create netcdf file with a time coordinate variable.

  # Create a time variable (using type POSIXct for convenience):
  nt <- 24
  time_posixct <- seq(ISOdatetime(1900,1,1,0,0,0,tz="UTC"), by="hour", len=nt)

  # Convert time variable to specified units:
```

```
  time_unit <- "hours since 1900-01-01 00:00:00 +00:00"
  time_coord <- utinvcal.nc(time_unit, time_posixct)

  # Create a netcdf file:
  file1 <- tempfile("utcal_", fileext=".nc")
  nc <- create.nc(file1)

  # Global attributes:
  att.put.nc(nc, "NC_GLOBAL", "Conventions", "NC_CHAR", "CF-1.6")
  att.put.nc(nc, "NC_GLOBAL", "title", "NC_CHAR", "RNetCDF example: time coordinate")
  att.put.nc(nc, "NC_GLOBAL", "institution", "NC_CHAR", "University of Areland")
  att.put.nc(nc, "NC_GLOBAL", "source", "NC_CHAR",
    paste("RNetCDF", utils::packageVersion("RNetCDF"), sep="_"))
  att.put.nc(nc, "NC_GLOBAL", "history", "NC_CHAR",
    paste(Sys.time(), "File created"))
  att.put.nc(nc, "NC_GLOBAL", "references", "NC_CHAR",
    "https://www.unidata.ucar.edu/software/udunits")
  att.put.nc(nc, "NC_GLOBAL", "comment", "NC_CHAR",
    "Uses attributes recommended by http://cfconventions.org")

  # Define time coordinate and attributes:
  dim.def.nc(nc, "time", nt)
  var.def.nc(nc, "time", "NC_DOUBLE", "time")
  att.put.nc(nc, "time", "long_name", "NC_CHAR", "time")
  att.put.nc(nc, "time", "units", "NC_CHAR", time_unit)
  # Calendar is optional (gregorian is the default):
  att.put.nc(nc, "time", "calendar", "NC_CHAR", "gregorian")

  # Write the data:
  var.put.nc(nc, "time", time_coord)

  close.nc(nc)

  ## Read time coordinate from netcdf file:

  # Open the file prepared earlier:
  nc <- open.nc(file1)

  # Read time coordinate and attributes:
  time_coord2 <- var.get.nc(nc, "time")
  time_unit2 <- att.get.nc(nc, "time", "units")

  close.nc(nc)
  unlink(file1)

  # Convert the time variable to POSIXct:
  time_posixct2 <- utcal.nc(time_unit2, time_coord2, "c")

  # Compare with original POSIXct variable:
  stopifnot(all.equal(time_posixct, time_posixct2))

}
```

utinit.nc *Initialize the UDUNITS2 Library*

Description

Initialize the UDUNITS2 library.

Usage

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

Arguments

path	Path to an XML-formatted unit-database for UDUNITS2.

Details

This function initializes the UDUNITS2 library. It is called by .onLoad when the package is loaded. Normally, the user does not need to call this function.

UDUNITS2 obtains a unit system by reading an XML file. The file name is given by argument path, if it is a non-empty character string. Otherwise, the file name is taken from environment variable UDUNITS2_XML_PATH, if it is a non-empty character string. By default, the file name is set to an XML file distributed with RNetCDF.

Author(s)

Pavel Michna, Milton Woods

References

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

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

Uses the UDUNITS library to convert a UTC-referenced date and time into the amount, value, of the temporal unit, unitstring.

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, Milton Woods

References

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

See Also

utcal.nc

Examples

```
if (inherits(try(utcal.nc("seconds since 1970-01-01", 0)), "try-error")) {
  warning("UDUNITS calendar conversions not supported by this build of RNetCDF")

} else {

  ##  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"))

}

##  An example of reading and writing a netcdf time coordinate
##  is given in the help for utcal.nc
```

var.def.nc *Define a NetCDF Variable*

Description

Define a new NetCDF variable.

Usage

```
var.def.nc(ncfile, varname, vartype, dimensions,
                  chunking=NA, chunksizes=NULL, deflate=NA, shuffle=FALSE,
                  big_endian=NA, fletcher32=FALSE,
                  filter_id=integer(0), filter_params=list())
```

Arguments

Arguments marked "netcdf4" are optional for datasets in that format and ignored for other formats.

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	External NetCDF data type as one of the following labels: NC_BYTE, NC_UBYTE, NC_CHAR, NC_SHORT, NC_USHORT, NC_INT, NC_UINT, NC_INT64, NC_UINT64, NC_FLOAT, NC_DOUBLE, NC_STRING, or a user-defined type name.
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.
chunking	("netcdf4") TRUE selects chunking, FALSE implies contiguous storage, NA allows the NetCDF library to choose a storage layout. Ignored for scalar variables.
chunksizes	("netcdf4") Chunk size expressed as the number of elements along each dimension, in the same order as dimensions. If NULL, the NetCDF library uses a default chunking strategy, which is intended to give reasonable performance in typical applications. Ignored unless chunking is TRUE.
deflate	("netcdf4") Integer indicating level of compression, from 0 (minimum) to 9 (maximum), or NA for no compression.
shuffle	("netcdf4") TRUE to enable byte shuffling, which may improve compression with deflate.
big_endian	("netcdf4") Byte order of the variable. TRUE for big-endian, FALSE for little-endian, NA for native endianness of the platform.
fletcher32	("netcdf4") TRUE to enable the fletcher32 checksum.

`filter_id` ("netcdf4") Vector of filter IDs to associate with the variable (empty vector denotes no filters). For information about the available filters, please see the NetCDF documentation. Ignored if the installed NetCDF library does not support the multi-filter interface.

`filter_params` ("netcdf4") List with one element for each `filter_id`. Each list member is a vector of `numeric` parameters (which are converted to unsigned integers). The meaning of the parameters depends on the filter implementation, and RNetCDF is unable to perform any validation. Ignored if the installed NetCDF library does not support the multi-filter interface.

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 fastest 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.

Value

NetCDF variable identifier, returned invisibly.

Author(s)

Pavel Michna, Milton Woods

References

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

Examples

```
##  Create a new NetCDF dataset and define two dimensions
file1 <- tempfile("var.def_", fileext=".nc")
nc <- create.nc(file1)

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)
unlink(file1)
```

`var.get.nc` *Read Data from a NetCDF Variable*

Description

Read the contents of a NetCDF variable.

Usage

```
var.get.nc(ncfile, variable, start=NA, count=NA,
  na.mode=4, collapse=TRUE, unpack=FALSE, rawchar=FALSE, fitnum=FALSE,
  cache_bytes=NA, cache_slots=NA, cache_preemption=NA)
```

Arguments

Arguments marked "netcdf4" are optional for datasets in that format and ignored for other formats.

`ncfile`	Object of class `NetCDF` which points to the NetCDF dataset (as returned from `open.nc`).
`variable`	ID or name of the NetCDF variable.
`start`	A vector of indices specifying the element where reading starts along each dimension of `variable`. Indices are numbered from 1 onwards, and the order of dimensions is shown by `print.nc` (array elements are stored sequentially with leftmost indices varying fastest). By default (`start=NA`), all dimensions of `variable` are read from the first element onwards. Otherwise, `start` must be a vector whose length is not less than the number of dimensions in `variable` (excess elements are ignored). Any `NA` values in vector `start` are set to 1.
`count`	A vector of integers specifying the number of values to read along each dimension of `variable`. The order of dimensions is the same as for `start`. By default (`count=NA`), all dimensions of `variable` are read from `start` to end. Otherwise, `count` must be a vector whose length is not less than the number of dimensions in `variable` (excess elements are ignored). Any `NA` value in vector `count` indicates that the corresponding dimension should be read from the `start` index to the end of the dimension.
`na.mode`	Missing values in the NetCDF dataset are converted to `NA` values in the result returned to R. The missing values are defined by attributes of the NetCDF `variable`, which are selected by the following modes:

mode	data type	attribute(s)
0	numeric	`_FillValue`, then `missing_value`
1	numeric	`_FillValue` only
2	numeric	`missing_value` only

3	any	no conversion
4	numeric	valid_range, valid_min, valid_max, _FillValue
5	any	same as mode 4 for numeric types; _FillValue for other types

For explanation of attribute conventions used by mode 4, please see: https://docs.unidata.ucar.edu/nug/current/attribute_conventions.html

`collapse` TRUE if degenerated dimensions (length=1) should be omitted.

`unpack` Packed variables are unpacked if unpack=TRUE and the attributes add_offset and/or 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.

`fitnum` By default, all numeric variables are read into R as double precision values. When fitnum==TRUE, the smallest R numeric type that can exactly represent each external type is used, as follows:

NC_BYTE	integer
NC_UBYTE	integer
NC_SHORT	integer
NC_USHORT	integer
NC_INT	integer
NC_UINT	double
NC_FLOAT	double
NC_DOUBLE	double
NC_INT64	integer64
NC_UINT64	integer64

`cache_bytes` ("netcdf4") Size of chunk cache in bytes. Value of NA (default) implies no change.

`cache_slots` ("netcdf4") Number of slots in chunk cache. Value of NA (default) implies no change.

`cache_preemption` ("netcdf4") Value between 0 and 1 (inclusive) that biases the cache scheme towards eviction of chunks that have been fully read. Value of NA (default) implies no change.

Details

NetCDF numeric variables cannot portably represent NA values from R. NetCDF does allow attributes to be defined for variables, and several conventions exist for attributes that define missing values and valid ranges. The convention in use can be specified by argument na.mode. Values of

a NetCDF variable that are deemed to be missing are automatically converted to NA in the results returned to R. Unusual cases can be handled directly in user code by setting na.mode=3.

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/or 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 variable is represented as a multi-dimensional array. The number and length of dimensions is determined when the variable is created. The start and count arguments of this routine indicate where the reading starts and the number of values to read along each dimension.

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

An array with dimensions determined by count and a data type that depends on the type of variable. For NetCDF variables of type NC_CHAR, the R type is either character or raw, as specified by argument rawchar. For NC_STRING, the R type is character. Numeric variables are read as double precision by default, but the smallest R type that exactly represents each external type is used if fitnum is TRUE.

Variables of user-defined types are supported. "compound" arrays are read into R as lists, with items named for the compound fields; items of base NetCDF data types are converted to R arrays, with leading dimensions from the field dimensions (if any) and trailing dimensions from the NetCDF variable. "enum" arrays are read into R as factor arrays. "opaque" arrays are read into R as raw (byte) arrays, with a leading dimension for bytes of the opaque type and trailing dimensions from the NetCDF variable. "vlen" arrays are read into R as a list with dimensions of the NetCDF variable; items in the list may have different lengths; base NetCDF data types are converted to R vectors.

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.

The arguments marked for "netcdf4" format refer to the chunk cache used for reading and writing variables. Default cache settings are defined by the NetCDF library, and they can be adjusted for each variable to improve performance in some applications.

Note

NC_BYTE is always interpreted as signed.

Author(s)

Pavel Michna, Milton Woods

References

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

Examples

```
##  Create a new NetCDF dataset and define two dimensions
file1 <- tempfile("var.get_", fileext=".nc")
nc <- create.nc(file1)

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 _FillValue attribute for temperature
att.put.nc(nc, "temperature", "_FillValue", "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(3,1), c(1,1))
var.get.nc(nc, "temperature", c(3,2))
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)
unlink(file1)
```

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, including its name, ID, type, number of dimensions, a vector of the dimension IDs, and the number of attributes.

Value

A list of named components, some of which are only included for datasets in "netcdf4" format (as reported by file.inq.nc).

id	Variable ID.
name	Variable name.
type	External NetCDF data type as one of the following labels: NC_BYTE, NC_UBYTE, NC_CHAR, NC_SHORT, NC_USHORT, NC_INT, NC_UINT, NC_INT64, NC_UINT64, NC_FLOAT, NC_DOUBLE, NC_STRING, or a user-defined type name.
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.
chunksizes	("netcdf4") Chunk size expressed as the number of elements along each dimension, in the same order as dimids. NULL implies contiguous storage.
cache_bytes	("netcdf4") Size of chunk cache in bytes (NULL if unsupported).
cache_slots	("netcdf4") The number of slots in the chunk cache (NULL if unsupported).
cache_preemption	("netcdf4") A value between 0 and 1 (inclusive) that biases the cache scheme towards eviction of chunks that have been fully read (NULL if unsupported).
deflate	("netcdf4") Integer indicating level of compression, from 0 (minimum) to 9 (maximum), or NA if compression is not enabled.

shuffle	("netcdf4") TRUE if byte shuffling is enabled for the variable, FALSE otherwise.
big_endian	("netcdf4") Byte order of the variable. TRUE for big-endian, FALSE for little-endian, NA for not yet determined, or NULL if unsupported.
fletcher32	("netcdf4") TRUE if the fletcher32 checksum is enabled for this variable, FALSE otherwise.
szip_options	("netcdf4") Integer containing a bitmask of szip options. NA if szip is not used, or NULL if unsupported.
szip_bits	("netcdf4") Number of bits per pixel for szip. NA if szip is not used, or NULL if unsupported.
filter_id	("netcdf4") Vector of filter IDs associated with the variable, or NULL if the NetCDF library does not support the multi-filter interface.
filter_params	("netcdf4") List with one element per filter_id, or NULL if the NetCDF library does not support the multi-filter interface. Each list member is a vector of numeric parameters for the corresponding filter. Please see the NetCDF documentation for information about the available filters and their parameters.

Author(s)

Pavel Michna, Milton Woods

References

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

Examples

```
##  Create a new NetCDF dataset and define two dimensions
file1 <- tempfile("var.inq_", fileext=".nc")
nc <- create.nc(file1)

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)
unlink(file1)
```

var.par.nc *Change Parallel Access Mode*

Description

Change the parallel access mode of a NetCDF variable from independent to collective and vice versa.

Usage

```
var.par.nc(ncfile, variable, access="NC_COLLECTIVE")
```

Arguments

ncfile	Object of class NetCDF which points to the NetCDF dataset (as returned from open.nc).
variable	Numeric ID or name of the variable for which to change the parallel access mode. Use "NC_GLOBAL" to change the parallel access mode for all variables in the dataset.
access	Parallel access mode as one of the following strings: "NC_COLLECTIVE" or "NC_INDEPENDENT".

Details

Parallel file access is either collective (all processors must participate) or independent (any processor may access the data without waiting for others). Data reads and writes (i.e. calls to var.put.nc and var.get.nc) are independent by default. Use this function to change the parallel access mode for a variable from independent to collective mode or vice versa.

All netCDF metadata writing operations are collective - all creation of groups, types, variables, dimensions, or attributes.

Note that when the file format is "classic" or "offset64", the change always applies to all variables in the file, even if a single variable is specified in argument variable.

Author(s)

Pavel Michna, Milton Woods

References

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

Examples

```
## Not run:
# This example assumes that the NetCDF library was built with MPI support,
# and that both RNetCDF and pbdMPI are installed in R.
# If the example code is stored in a file myexample.R,
# run R under MPI using a command similar to:
# SHELL> mpiexec -np 2 Rscript --vanilla myexample.R

library(pbdMPI, quiet = TRUE)
library(RNetCDF, quiet = TRUE)

# Get MPI parameters
init()
rank <- comm.rank()
size <- comm.size()

# Define dimensions and data
nr <- 5
nc_local <- 4
nc <- nc_local * size
data_local <- matrix(rank, nrow=nr, ncol=nc_local)

# Open file for parallel access and define metadata
filename <- "myexample.nc"
info.create()
ncid <- create.nc(filename, format="netcdf4", mpi_comm=comm.c2f(), mpi_info=info.c2f())
rdim <- dim.def.nc(ncid, "rows", nr)
cdim <- dim.def.nc(ncid, "cols", nc)
varid <- var.def.nc(ncid, "data", "NC_INT", c(rdim, cdim))

# Use collective I/O
var.par.nc(ncid, "data", "NC_COLLECTIVE")

# Write data
var.put.nc(ncid, varid, data_local, start=c(1,rank*nc_local+1), count=c(nr,nc_local))

# Finish up
close.nc(ncid)
info.free()

finalize()

## End(Not run)
```

`var.put.nc` *Write Data to a NetCDF Variable*

Description

Write the contents of a NetCDF variable.

Usage

```
var.put.nc(ncfile, variable, data, start=NA, count=NA, na.mode=4, pack=FALSE,
  cache_bytes=NA, cache_slots=NA, cache_preemption=NA)
```

Arguments

Arguments marked "netcdf4" are optional for datasets in that format and ignored for other formats.

ncfile	Object of class NetCDF which points to the NetCDF dataset (as returned from open.nc).
variable	ID or name of the NetCDF variable.
data	An R vector or array of data to be written to the NetCDF variable. Values are taken from data in the order of R vector elements, so that leftmost indices vary fastest over an array.
start	A vector of indices specifying the element where writing starts along each dimension of variable. Indices are numbered from 1 onwards, and the order of dimensions is shown by print.nc (array elements are stored sequentially with leftmost indices varying fastest). By default (start=NA), all dimensions of variable are written from the first element onwards. Otherwise, start must be a vector whose length is not less than the number of dimensions in variable (excess elements are ignored). Any NA values in vector start are set to 1.
count	A vector of integers specifying the number of values to write along each dimension of variable. The order of dimensions is the same as for start. By default (count=NA), count is set to dim(data) for an array or length(data) for a vector. Otherwise, count must be a vector whose length is not less than the number of dimensions in variable (excess elements are ignored). Any NA value in vector count indicates that the corresponding dimension should be written from the start index to the end of the dimension. Note that an unlimited dimension initially has zero length, and the dimension is extended by setting the corresponding element of count greater than the current length.
na.mode	NA values in data are converted to a missing value in the NetCDF dataset. The missing value is defined by attributes of the NetCDF variable, which are selected by the following modes:

mode	data type	attribute(s)
0	numeric	_FillValue, then missing_value
1	numeric	_FillValue only
2	numeric	missing_value only
3	any	no conversion
4	numeric	valid_range valid_min, valid_max, _FillValue
5	any	same as mode 4 for numeric types; _FillValue for other types

For explanation of attribute conventions used by mode 4, please see: https://docs.unidata.ucar.edu/nug/current/attribute_conventions.html

pack — Variables are packed if pack=TRUE and the attributes add_offset and/or scale_factor are defined. Default is FALSE.

cache_bytes — ("netcdf4") Size of chunk cache in bytes. Value of NA (default) implies no change.

cache_slots — ("netcdf4") Number of slots in chunk cache. Value of NA (default) implies no change.

cache_preemption — ("netcdf4") Value between 0 and 1 (inclusive) that biases the cache scheme towards eviction of chunks that have been fully read. Value of NA (default) implies no change.

Details

This function writes values to a NetCDF variable. Data values in R are automatically converted to the correct type of NetCDF variable.

Text represented by R type character can be written to NetCDF types NC_CHAR and NC_STRING, and R type raw can be written to NetCDF type NC_CHAR. When writing to NC_CHAR variables, 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.

Due to the lack of native support for 64-bit integers in R, NetCDF types NC_INT64 and NC_UINT64 require special attention. This function accepts the usual R integer (signed 32-bit) and numeric (double precision) types, but to represent integers larger than about 53-bits without truncation, integer64 vectors are also supported.

NetCDF numeric variables cannot portably represent NA values from R. NetCDF does allow attributes to be defined for variables, and several conventions exist for attributes that define missing values and valid ranges. The convention in use can be specified by argument na.mode. Values of NA in argument data are converted to a missing or fill value before writing to the NetCDF variable. Unusual cases can be handled directly in user code by setting na.mode=3.

Variables of user-defined types are supported, subject to conditions on the corresponding data structures in R. "compound" arrays must be stored in R as lists, with items named for the compound fields; items of base NetCDF data types are stored as R arrays, with leading dimensions from the field dimensions (if any) and trailing dimensions from the NetCDF variable. "enum" arrays are stored in R as factor arrays. "opaque" arrays are stored in R as raw (byte) arrays, with a leading dimension for bytes of the opaque type and trailing dimensions from the NetCDF variable. "vlen" arrays are stored in R as a list with dimensions of the NetCDF variable; items in the list may have different lengths; base NetCDF data types are stored as R vectors.

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 attributes add_offset and/or 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 variable is represented as a multi-dimensional array. The number and length of dimensions is determined when the variable is created. The start and count arguments of this routine indicate where the writing starts and the number of values to write along each dimension.

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).

The arguments marked for `"netcdf4"` format refer to the chunk cache used for reading and writing variables. Default cache settings are defined by the NetCDF library, and they can be adjusted for each variable to improve performance in some applications.

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

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

Examples

```
##  Create a new NetCDF dataset and define two dimensions
file1 <- tempfile("var.put_", fileext=".nc")
nc <- create.nc(file1)

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 _FillValue attribute for temperature
att.put.nc(nc, "temperature", "_FillValue", "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 subsets of the data:
var.put.nc(nc, "time", mytime, start=2, count=1)
var.put.nc(nc, "temperature", mytemperature[3:4,2], start=c(3,2), count=c(2,1))
var.put.nc(nc, "name", myname[3:4], start=c(NA,3), count=c(NA,2))
sync.nc(nc)
```

```
##  Put all of the data:
var.put.nc(nc, "time", mytime)
var.put.nc(nc, "temperature", mytemperature)
var.put.nc(nc, "name", myname)

close.nc(nc)
unlink(file1)
```

`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, Milton Woods

References

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

Examples

```
##  Create a new NetCDF dataset and define two dimensions
file1 <- tempfile("var.rename_", fileext=".nc")
nc <- create.nc(file1)

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)
unlink(file1)
```

Index