Octave NetCDF Toolkit 1.0.17

NetCDF functions for GNU Octave.

John Donoghue

Distribution

The GNU Octave NetCDF package is *free* software. Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software. This means that everyone is free to use it and free to redistribute it on certain conditions. The GNU Octave NetCDF package is not, however, in the public domain. It is copyrighted and there are restrictions on its distribution, but the restrictions are designed to ensure that others will have the same freedom to use and redistribute Octave that you have. The precise conditions can be found in the GNU General Public License that comes with the GNU Octave NetCDF package and that also appears in Appendix A [Copying], page 23.

To download a copy of the GNU Octave NetCDF package, please visit `http://octave.sourceforge.net/netcdf/`.

Table of Contents

1 Installing and loading

The toolkit must be installed and then loaded to be used.

It can be installed in GNU Octave directly from the website, or can be installed in an off-line mode via a downloaded tarball.

The toolkit has a dependency on the netcdf library (https://www.unidata.ucar.edu/software/netcdf/), so it must be installed in order to successfully install the toolkit.

The toolkit must be then be loaded once per each GNU Octave session in order to use its functionality.

1.1 Online Direct install

With an internet connection available, the package can be installed from octave-forge using the following command within GNU Octave:

```
pkg install -forge netcdf
```

The latest released version of the toolkit will be downloaded and installed.

1.2 Off-line install

With the toolkit package already downloaded, and in the current directory when running GNU Octave, the package can be installed using the following command within GNU Octave:

```
pkg install netcdf-1.0.17.tar.gz
```

1.3 Loading

Regardless of the method of installing the toolkit, in order to use its functions, the toolkit must be loaded using the pkg load command:

```
pkg load netcdf
```

The toolkit must be loaded on each GNU Octave session.

2 Basic Usage Overview

The toolkit provides high and level functionality for reading and writing NetCDF format files.

2.1 High level functionality

The toolkit provides the following high level functions:

- nccreate
- ncdisp
- ncinfo
- ncreadatt
- ncread
- ncwriteatt
- ncwrite
- ncwriteschema

2.2 Low level functionality

The package aims to implement the netcdf interface of MATLAB in GNU Octave, however GNU Octave does not support the import function.

Functions can be used in netcdf_functionname format, or an emulated import can be done using the import_netcdf script so that functions can be used in netcdf.functionname format.

3 Function Reference

The functions currently available in the toolkit are described below;

3.1 High-level functions

3.1.1 nccreate

`nccreate(`*`filename`*`,`*`varname`*`)` [Function File]
`nccreate(`*`filename`*`,`*`varname`*`,"property",`*`value`*`,...)` [Function File]
Create the variable *varname* in the file *filename*.

Properties

The following properties can be used:

- "Dimensions": a cell array with the dimension names followed by their length or Inf if the dimension is unlimited. If the property is omitted, a scalar variable is created.
- "Datatype": a string with the Octave data type name (see `ncinfo` for the correspondence between Octave and NetCDF data types). The default data type is a "double".
- "Format": This can be "netcdf4_classic" (default), "classic", "64bit" or "netcdf4".
- "FillValue": the value used for undefined elements of the NetCDF variable.
- "ChunkSize": the size of the data chunks. If omitted, the variable is not chunked.
- "DeflateLevel": The deflate level for compression. It can be the string "disable" (default) for no compression or an integer between 0 (no compression) and 9 (maximum compression).
- "Shuffle": true for enabling the shuffle filter or false (default) for disabling it.

Example

```
nccreate("test.nc","temp","Dimensions",{"lon",10,"lat",20},"Format","classic");
ncdisp("test.nc");
```

See also: ncwrite.

3.1.2 ncdisp

`ncdisp (`*`filename`*`)` [Function File]
Display meta-data of the NetCDF file *filename*

Example

```
ncdisp("test.nc");
```

See also: ncinfo.

3.1.3 ncinfo

`info` `= ncinfo (`*`filename`*`)` [Function File]
`info` `= ncinfo (`*`filename`*`, `*`varname`*`)` [Function File]
`info` `= ncinfo (`*`filename`*`, `*`groupname`*`)` [Function File]
Return information about an entire NetCDF file *filename* (i.e. the root group "/"), about the variable called *varname* or the group called *groupname*.

The structure *info* has always the following fields:

- *Filename*: the name of the NetCDF file

- *Format*: one of the strings "CLASSIC", "64BIT", "NETCDF4" or "NETCDF4_CLASSIC"

The structure *info* has additional fields depending on whether a group of variable is queried.

Groups

Groups are returned as an array structure with the following fields:

- *Name*: the group name. The root group is named "/".
- *Dimensions*: a array structure with the dimensions.
- *Variables*: a array structure with the variables.
- *Attributes*: a array structure with global attributes.
- *Groups*: a array structure (one for each group) with the same fields as this structure.

Dimensions

Dimensions are returned as an array structure with the following fields:

- *Name*: the name of the dimension
- *Length*: the length of the dimension
- *Unlimited*: true of the dimension has no fixed limited, false

Variables

Variables are returned as an array structure with the following fields:

- *Name*: the name of the dimension
- *Dimensions*: array structure of all dimensions of this variable with the same structure as above.
- *Size*: array with the size of the variable
- *Datatype*: string with the corresponding octave data-type (see below)
- *Attributes*: a array structure of attributes
- *FillValue*: the NetCDF fill value of the variable. If the fill value is not defined, then this attribute is an empty array ([]).
- *DeflateLevel*: the NetCDF deflate level between 0 (no compression) and 9 (maximum compression).
- *Shuffle*: is true if the shuffle filter is activated to improve compression, otherwise false.
- *CheckSum*: is set to "fletcher32", if check-sums are used, otherwise this field is not defined.

Attributes

Attributes are returned as an array structure with the following fields:

- *Name*: the name of the attribute
- *Value*: the value of the attribute (with the corresponding type)
- *Unlimited*: true of the dimension has no fixed limited, false

Data-types

The following the the correspondence between the Octave and NetCDF data-types:

Octave type	**NetCDF type**
`int8`	`NC_BYTE`
`uint8`	`NC_UBYTE`
`int16`	`NC_SHORT`

uint16	NC_USHORT
int32	NC_INT
uint32	NC_UINT
int64	NC_INT64
uint64	NC_UINT64
single	NC_FLOAT
double	NC_DOUBLE
char	NC_CHAR

The output of `ncinfo` can be used to create a NetCDF file with the same meta-data using `ncwriteschema`.

Note: If there are no attributes (or variable or groups), the corresponding field is an empty matrix and not an empty struct array for compatibility with matlab.

See also: ncread,nccreate,ncwriteschema,ncdisp.

3.1.4 ncread

`x = ncread (filename, varname)` [Function File]
`x = ncread (filename, varname,start,count,stride)` [Function File]

Read the variable *varname* from the NetCDF file *filename*.

If *start*,*count* and *stride* are present, a subset of the variable is loaded. The parameter *start* contains the starting indices (1-based), *count* is the number of elements and *stride* the increment between two successive elements. These parameters are vectors whose length is equal to the number of dimension of the variable. Elements of *count* might be Inf which means that as many values as possible are loaded.

If the variable has the _FillValue attribute, then the corresponding values are replaced by NaN (except for characters). NetCDF attributes scale_factor (default 1) and add_offset (default 0) are use the transform the variable during the loading:

x = scale_factor * x_in_file + add_offset

The output data type matches the NetCDF datatype, except when the attributes _FillValue, add_offset or scale_factor are defined in which case the output is a array in double precision.

Note that values equal to the attribute missing_value are not replaced by NaN (for compatibility).

Example

Read the data from variable 'mydata' in the file test.nc.

```
data  = ncread('test.nc','mydata');
```

See also: ncwrite,ncinfo,ncdisp.

3.1.5 ncreadatt

`val = ncreadatt(filename,varname,attname)` [Function File]

Return the attribute *attname* of the variable *varname* in the file *filename*.

Global attributes can be accessed by using `"/"` or the group name as *varname*. The type of attribute is mapped to the Octave data types. (see `ncinfo`).

Example

Read global attribute 'creation_date'

```
d = ncreadatt('test.nc','/','creation_date')
```

Read atribute 'myattr' assigned to variable mydata.

```
d = ncreadattr('test.nc', 'mydata', 'myattr');
```

See also: ncinfo,ncwriteatt.

3.1.6 ncwrite

`ncwrite (filename, varname, x)` [Function File]
`ncwrite (filename, varname, x, start, stride)` [Function File]

Write array *x* to the the variable *varname* in the NetCDF file *filename*.

The variable with the name *varname* and the appropriate dimension must already exist in the NetCDF file.

If *start* and *stride* are present, a subset of the variable is written. The parameter *start* contains the starting indices (1-based) and *stride* the increment between two successive elements. These parameters are vectors whose length is equal to the number of dimension of the variable.

If the variable has the _FillValue attribute, then the values equal to NaN are replaced by corresponding fill value NetCDF attributes scale_factor (default 1) and add_oddset (default 0) are use the transform the variable during writing:

x_in_file = (x - add_offset)/scale_factor

Example

Create a netcdf file with a variable of 'mydata' and then write data to that variable.

```
nccreate('myfile.nc','mydata');
ncwrite('myfile.nc','mydata', 101);
```

See also: ncread,nccreate.

3.1.7 ncwriteatt

`ncwriteatt(filename,varname,attname,val)` [Function File]

Defines the attribute *attname* of the variable *varname* in the file *filename* with the value *val*.

Global attributes can be defined by using `"/"` or the group name as *varname*. The type of value is mapped to the NetCDF data types. (see `ncinfo`).

Example

Create a netcdf4 format file with a variable mydata and assign an attribute "units" to it.

```
nccreate("myfile.nc", "mydata", "Format", "netcdf4");
ncwriteatt("myfile.nc", "mydata", "Units", "K");
```

See also: ncinfo.

3.1.8 ncwriteschema

`ncwriteschema (filename, schema)` [Function File]

Create a NetCDF called *filename* with the dimensions, attributes, variables and groups given by the structure *schema*.

The variable *schema* has the same structure as the results of `ncinfo`. `ncinfo` and `ncwriteschema` can be used together to create a NetCDF using another file as a template:

Example

```
schema = ncinfo("template.nc");
# the new file should be named "new_file.nc"
ncwriteschema("new_file.nc",schema);
```

Unused field in *schema* such as *ChunkSize*, *Shuffle*, *DeflateLevel*, *FillValue*, *Checksum* can be left-out if the corresponding feature is not used.

Dimensions are considered as limited if the field *Unlimited* is missing, unless the dimension length is Inf.

See also: ncinfo.

3.2 Low-level functions (Deprecated)

3.2.1 netcdf_abort

`netcdf_abort(`*ncid*`)` [Loadable Function]
Aborts all changes since the last time the dataset entered in define mode.

See also: netcdf_reDef.

3.2.2 netcdf_close

`netcdf_close(`*ncid*`)` [Loadable Function]
Close the NetCDF file with the id *ncid*.

See also: netcdf_open.

3.2.3 netcdf_copyAtt

`netcdf_copyAtt (`*ncid*`,`*varid*`,`*name*`,`*ncid_out*`,`*varid_out*`)` [Loadable Function]
Copies the attribute named *old_name* of the variable *varid* in the data set *ncid* to the variable *varid_out* in the data set *ncid_out*. To copy a global attribute use netcdf_getConstant("global") for *varid* or *varid_out*.

See also: netcdf_getAtt,netcdf_getConstant.

3.2.4 netcdf_create

ncid `= netcdf_create(`*filename*`,`*mode*`)` [Loadable Function]
Creates the file named *filename* in the mode *mode* which can have the following values: "clobber" (overwrite existing files), "noclobber" (prevent to overwrite existing files) "64bit_offset" (use the 64bit-offset format), "netcdf4" (use the NetCDF4, i.e. HDF5 format) or "share" (concurrent reading of the dataset). *mode* can also be the numeric value return by netcdf_getConstant. In the later-case it can be combined with a bitwise-or.

Example

```
mode =  bitor(netcdf.getConstant("classic_model"), ...
netcdf.getConstant("netcdf4"));
ncid = netcdf.create("test.nc",mode);
```

See also: netcdf_close.

3.2.5 netcdf_defDim

dimid `= netcdf_defDim(`*ncid*`,`*name*`,`*len*`)` [Loadable Function]
Define the dimension with the name *name* and the length *len* in the dataset *ncid*. The id of the dimension is returned.

See also: netcdf_defVar.

3.2.6 netcdf_defGrp

new_ncid = netcdf_defGrp(*ncid*,*name*) [Loadable Function]

Define a group in a NetCDF file.

See also: netcdf_inqGrps.

3.2.7 netcdf_defVar

varid = netcdf_defVar(*ncid*,*name*,*xtype*,*dimids*) [Loadable Function]

Defines a variable with the name *name* in the dataset *ncid*. *xtype* can be "byte", "ubyte", "short", "ushort", "int", "uint", "int64", "uint64", "float", "double", "char" or the corresponding number as returned by netcdf_getConstant. The parameter *dimids* define the ids of the dimension. For scalar this parameter is the empty array ([]). The variable id is returned.

See also: netcdf_open,netcdf_defDim.

3.2.8 netcdf_defVarChunking

netcdf_defVarChunking (*ncid*,*varid*,*storage*,*chunkSizes*) [Loadable Function]

Define the chunking settings of NetCDF variable *varid*. If *storage* is the string "chunked", the variable is stored by chunk of the size *chunkSizes*. If *storage* is the string "contiguous", the variable is stored in a contiguous way.

See also: netcdf_inqVarChunking.

3.2.9 netcdf_defVarDeflate

netcdf_defVarDeflate [Loadable Function]
(*ncid*,*varid*,*shuffle*,*deflate*,*deflate_level*)

Define the compression settings NetCDF variable *varid*. If *deflate* is true, then the variable is compressed. The compression level *deflate_level* is an integer between 0 (no compression) and 9 (maximum compression).

See also: netcdf_inqVarDeflate.

3.2.10 netcdf_defVarFill

netcdf_defVarFill(*ncid*,*varid*,*no_fill*,*fillvalue*) [Loadable Function]

Define the fill-value settings of the NetCDF variable *varid*. If *no_fill* is false, then the values between no-contiguous writes are filled with the value *fill_value*. This is disabled by setting *no_fill* to true.

See also: netcdf_inqVarFill.

3.2.11 netcdf_defVarFletcher32

netcdf_defVarFletcher32(*ncid*,*varid*,*checksum*) [Loadable Function]

Defines the checksum settings of the variable with the id *varid* in the data set *ncid*. If *checksum* is the string "FLETCHER32", then fletcher32 checksums will be turned on for this variable. If *checksum* is "NOCHECKSUM", then checksums will be disabled.

See also: netcdf_defVar,netcdf_inqVarFletcher32.

3.2.12 netcdf_defVlen

varid = `netcdf_defVlen(`*ncid*`,`*typename*`,`*basetype*`)` [Loadable Function]
Defines a NC_VLEN variable length array type with the type name *typename* and a base datatype of *basetype* in the dataset *ncid*. *basetype* can be "byte", "ubyte", "short", "ushort", "int", "uint", "int64", "uint64", "float", "double", "char" or the corresponding number as returned by netcdf_getConstant. The new data type id is returned.

See also: netcdf_open,netcdf_defVar, netcdf_inqVlen.

3.2.13 netcdf_delAtt

`netcdf_delAtt(`*ncid*`,`*varid*`,`*name*`)` [Loadable Function]
Deletes the attribute named *name* of the variable *varid* in the data set *ncid*. To delete a global attribute use netcdf_getConstant("global") for *varid*.

See also: netcdf_defAtt,netcdf_getConstant.

3.2.14 netcdf_endDef

`netcdf_endDef (`*ncid*`)` [Loadable Function]
Leaves define-mode of NetCDF file *ncid*.

See also: netcdf_reDef.

3.2.15 netcdf_getAtt

data = `netcdf_getAtt (`*ncid*`,`*varid*`,`*name*`)` [Loadable Function]
Get the value of a NetCDF attribute. This function returns the value of the attribute called *name* of the variable *varid* in the NetCDF file *ncid*. For global attributes *varid* can be netcdf_getConstant("global").

See also: netcdf_putAtt.

3.2.16 netcdf_getChunkCache

[*size*, *nelems*, *preemption*] = `netcdf_getChunkCache()` [Loadable Function]
Gets the default chunk cache settings in the HDF5 library.

See also: netcdf_setChunkCache.

3.2.17 netcdf_getConstant

value = `netcdf_getConstant(`*name*`)` [Loadable Function]
Returns the value of a NetCDF constant called *name*.

See also: netcdf_getConstantNames.

3.2.18 netcdf_getConstantNames

value = `netcdf_getConstantNames()` [Loadable Function]
Returns a list of all constant names.

See also: netcdf_getConstant.

3.2.19 netcdf_getVar

data = netcdf_getVar (*ncid*,*varid*) [Loadable Function]
data = netcdf_getVar (*ncid*,*varid*,*start*) [Loadable Function]
data = netcdf_getVar (*ncid*,*varid*,*start*,*count*) [Loadable Function]
data = netcdf_getVar (*ncid*,*varid*,*start*,*count*,*stride*) [Loadable Function]
Get the data from a NetCDF variable. The data *data* is loaded from the variable *varid* of the NetCDF file *ncid*. *start* is the start index of each dimension (0-based and defaults to a vector of zeros), *count* is the number of elements of to be written along each dimension (default all elements) and *stride* is the sampling interval.

See also: netcdf_putVar.

3.2.20 netcdf_inq

vers = netcdf_inqLibVers() [Loadable Function]
Returns the version of the NetCDF library.

See also: netcdf_open.

3.2.21 netcdf_inqAtt

name = netcdf_inqAttName (*ncid*,*varid*,*attnum*) [Loadable Function]
Get the name of a NetCDF attribute. This function returns the name of the attribute with the id *attnum* of the variable *varid* in the NetCDF file *ncid*. For global attributes *varid* can be netcdf_getConstant("global").

See also: netcdf_inqAttName.

3.2.22 netcdf_inqAttID

attnum = netcdf_inqAttID(*ncid*,*varid*,*attname*) [Loadable Function]
Return the attribute id *attnum* of the attribute named *attname* of the variable *varid* in the dataset *ncid*. For global attributes *varid* can be netcdf_getConstant("global").

See also: netcdf_inqAttName.

3.2.23 netcdf_inqAttName

name = netcdf_inqAttName (*ncid*,*varid*,*attnum*) [Loadable Function]
Get the name of a NetCDF attribute. This function returns the name of the attribute with the id *attnum* of the variable *varid* in the NetCDF file *ncid*. For global attributes *varid* can be netcdf_getConstant("global").

See also: netcdf_inqAttName.

3.2.24 netcdf_inqDim

[*name*,*length*] = netcdf_inqDim(*ncid*,*dimid*) [Loadable Function]
Returns the name and length of a NetCDF dimension.

See also: netcdf_inqDimID.

3.2.25 netcdf_inqDimID

dimid = netcdf_inqDimID(*ncid*,*dimname*) [Loadable Function]
Return the id of a NetCDF dimension.

See also: netcdf_inqDim.

3.2.26 netcdf_inqDimIDs

dimids = netcdf_inqDimID(*ncid*) [Loadable Function]
dimids = netcdf_inqDimID(*ncid*,*include_parents*) [Loadable Function]
Return the dimension ids defined in a NetCDF file. If *include_parents* is 1, the dimension ids of the parent group are also returned. Per default this is not the case (*include_parents* is 0).

See also: netcdf_inqDim.

3.2.27 netcdf_inqFormat

format = netcdf_inqFormat(*ncid*) [Loadable Function]
Return the NetCDF format of the dataset *ncid*. Format might be one of the following "FORMAT_CLASSIC", "FORMAT_64BIT", "FORMAT_NETCDF4" or "FORMAT_NETCDF4_CLASSIC"

See also: netcdf_inq.

3.2.28 netcdf_inqGrpFullNcid

grp_ncid = netcdf_inqGrpFullNcid(*ncid*,*name*) [Loadable Function]
Return the group id based on the full group name.

See also: netcdf_inqGrpName.

3.2.29 netcdf_inqGrpName

name = netcdf_inqGrpName(*ncid*) [Loadable Function]
Return group name in a NetCDF file.

See also: netcdf_inqGrps.

3.2.30 netcdf_inqGrpNameFull

name = netcdf_inqGrpNameFull(*ncid*) [Loadable Function]
Return full name of group in NetCDF file.

See also: netcdf_inqGrpName.

3.2.31 netcdf_inqGrpParent

parent_ncid = netcdf_inqGrpParent(*ncid*) [Loadable Function]
Return id of the parent group

See also: netcdf_inqGrpName.

3.2.32 netcdf_inqGrps

ncids = netcdf_inqGrps(*ncid*) [Loadable Function]
Return all groups ids in a NetCDF file.

See also: netcdf_inqGrps.

3.2.33 netcdf_inqLibVers

vers = netcdf_inqLibVers() [Loadable Function]
Returns the version of the NetCDF library.

See also: netcdf_open.

3.2.34 netcdf_inqNcid

grp_ncid = netcdf_inqNcid(*ncid*,*name*) [Loadable Function]
Return group id based on its name

See also: netcdf_inqGrpFullNcid.

3.2.35 netcdf_inqUnlimDims

unlimdimids = netcdf_inqUnlimDims(*ncid*) [Loadable Function]
Return the id of all unlimited dimensions of the NetCDF file *ncid*.

See also: netcdf_inq.

3.2.36 netcdf_inqUserType

[*typename*, *bytesize*, *basetypeid*, *numfields*, *classid*] = [Loadable Function]
netcdf_inqUserType(*ncid*,*typeid*)
Provide information on a user defined type *typeid* in the dataset *ncid*.

The function returns the typename, bytesize, base type id, number of fields and class identifier of the type.

See also: netcdf_open, netcdf_defVlen, netcdf_inqVlen.

3.2.37 netcdf_inqVar

[*no_fill*,*fillvalue*] = netcdf_inqVarFill(*ncid*,*varid*) [Loadable Function]
Determines the fill-value settings of the NetCDF variable *varid*. If *no_fill* is false, then the values between no-contiguous writes are filled with the value *fill_value*. This is disabled by setting *no_fill* to true.

See also: netcdf_defVarFill.

3.2.38 netcdf_inqVarChunking

[*storage*,*chunkSizes*] = netcdf_inqVarChunking [Loadable Function]
(*ncid*,*varid*)
Determines the chunking settings of NetCDF variable *varid*. If *storage* is the string "chunked", the variable is stored by chunk of the size *chunkSizes*. If *storage* is the string "contiguous", the variable is stored in a contiguous way.

See also: netcdf_defVarChunking.

3.2.39 netcdf_inqVarDeflate

[*shuffle*,*deflate*,*deflate_level*] = [Loadable Function]
netcdf_inqVarDeflate (*ncid*,*varid*)
Determines the compression settings NetCDF variable *varid*. If *deflate* is true, then the variable is compressed. The compression level *deflate_level* is an integer between 0 (no compression) and 9 (maximum compression).

See also: netcdf_defVarDeflate.

3.2.40 netcdf_inqVarFill

`[no_fill,fillvalue] = netcdf_inqVarFill(ncid,varid)` [Loadable Function]
Determines the fill-value settings of the NetCDF variable *varid*. If *no_fill* is false, then the values between no-contiguous writes are filled with the value *fill_value*. This is disabled by setting *no_fill* to true.

See also: netcdf_defVarFill.

3.2.41 netcdf_inqVarFletcher32

`checksum = netcdf_inqVarFletcher32(ncid,varid)` [Loadable Function]
Determines the checksum settings of the variable with the id *varid* in the data set *ncid*. If fletcher32 checksums is turned on for this variable, then *checksum* is the string "FLETCHER32". Otherwise it is the string "NOCHECKSUM".

See also: netcdf_defVar,netcdf_inqVarFletcher32.

3.2.42 netcdf_inqVarID

`varid = netcdf_inqVarID (ncid,name)` [Loadable Function]
Return the id of a variable based on its name.

See also: netcdf_defVar,netcdf_inqVarIDs.

3.2.43 netcdf_inqVarIDs

`varids = netcdf_inqVarID (ncid)` [Loadable Function]
Return all variable ids. This functions returns all variable ids in a NetCDF file or NetCDF group.

See also: netcdf_inqVarID.

3.2.44 netcdf_inqVlen

`[typename, bytesize, basetypeid] = netcdf_inqVlen(ncid,typeid)` [Loadable Function]
Provide information on a NC_VLEN variable length array type *typeid* in the dataset *ncid*.

The function returns the typename, bytesize, and base type id.

See also: netcdf_open, netcdf_defVlen.

3.2.45 netcdf_open

`ncid = netcdf_open(filename,mode)` [Loadable Function]
Opens the file named *filename* in the mode *mode*.

See also: netcdf_close.

3.2.46 netcdf_putAtt

`netcdf_putAtt (ncid,varid,name,data)` [Loadable Function]
Defines a NetCDF attribute. This function defines the attribute called *name* of the variable *varid* in the NetCDF file *ncid*. The value of the attribute will be *data*. For global attributes *varid* can be netcdf_getConstant("global").

See also: netcdf_getAtt.

3.2.47 netcdf_putVar

`netcdf_putVar (ncid,varid,data)` [Loadable Function]
`netcdf_putVar (ncid,varid,start,data)` [Loadable Function]
`netcdf_putVar (ncid,varid,start,count,data)` [Loadable Function]
`netcdf_putVar (ncid,varid,start,count,stride,data)` [Loadable Function]
Put data in a NetCDF variable. The data *data* is stored in the variable *varid* of the NetCDF file *ncid*. *start* is the start index of each dimension (0-based and defaults to a vector of zeros), *count* is the number of elements of to be written along each dimension (default all elements) and *stride* is the sampling interval.

See also: netcdf_endDef.

3.2.48 netcdf_reDef

`netcdf_reDef (ncid)` [Loadable Function]
Enter define-mode of NetCDF file *ncid*.

See also: netcdf_endDef.

3.2.49 netcdf_renameAtt

`netcdf_renameAtt(ncid,varid,old_name,new_name)` [Loadable Function]
Renames the attribute named *old_name* of the variable *varid* in the data set *ncid*. *new_name* is the new name of the attribute. To rename a global attribute use netcdf_getConstant("global") for *varid*.

See also: netcdf_copyAtt,netcdf_getConstant.

3.2.50 netcdf_renameDim

`netcdf_renameDim(ncid,dimid,name)` [Loadable Function]
Renames the dimension with the id *dimid* in the data set *ncid*. *name* is the new name of the dimension.

See also: netcdf_defDim.

3.2.51 netcdf_renameVar

`netcdf_renameVar(ncid,varid,name)` [Loadable Function]
Renames the variable with the id *varid* in the data set *ncid*. *name* is the new name of the variable.

See also: netcdf_defVar.

3.2.52 netcdf_setChunkCache

`netcdf_setChunkCache(size, nelems, preemption)` [Loadable Function]
Sets the default chunk cache settings in the HDF5 library. The settings applies to all files which are subsequently opened or created.

See also: netcdf_getChunkCache.

3.2.53 netcdf_setDefaultFormat

old_format = `netcdf_setDefaultFormat(`*format*`)` [Loadable Function]
Sets the default format of the NetCDF library and returns the previous default format (as a numeric value). *format* can be "format_classic", "format_64bit", "format_netcdf4" or "format_netcdf4_classic".

See also: netcdf_open.

3.2.54 netcdf_setFill

old_mode = `netcdf_setFill(`*ncid*`,`*fillmode*`)` [Loadable Function]
Change the fill mode (*fillmode*) of the data set *ncid*. The previous value of the fill mode is returned. *fillmode* can be either "fill" or "nofill".

See also: netcdf_open.

3.2.55 netcdf_sync

`netcdf_sync(`*ncid*`)` [Loadable Function]
Writes all changes to the disk and leaves the file open.

See also: netcdf_close.

3.3 Low-level functions

3.3.1 netcdf.abort

`netcdf.abort(`*ncid*`)`
Aborts all changes since the last time the dataset entered in define mode.

3.3.2 netcdf.close

`netcdf.close(`*ncid*`)`
Close the NetCDF file with the id *ncid*.

3.3.3 netcdf.copyAtt

`netcdf.copyAtt (`*ncid*`,`*varid*`,`*name*`,`*ncid_out*`,`*varid_out*`)`
Copies the attribute named *old_name* of the variable *varid* in the data set *ncid* to the variable *varid_out* in the data set *ncid_out*. To copy a global attribute use netcdf.getConstant("global") for *varid* or *varid_out*.

See also: netcdf.getAtt,netcdf.getConstant.

3.3.4 netcdf.create

ncid = `netcdf.create(`*filename*`,`*mode*`)`
Creates the file named *filename* in the mode *mode* which can have the following values: "clobber" (overwrite existing files), "noclobber" (prevent to overwrite existing files) "64bit_offset" (use the 64bit-offset format), "netcdf4" (use the NetCDF4, i.e. HDF5 format) or "share" (concurrent reading of the dataset). *mode* can also be the numeric value return by netcdf.getConstant. In the later-case it can be combined with a bitwise-or.

Example

```
mode = bitor(netcdf.getConstant("classic_model"), ...
netcdf.getConstant("netcdf4"));
ncid = netcdf.create("test.nc",mode);
```

3.3.5 netcdf.defDim

dimid = netcdf.defDim(*ncid*,*name*,*len*)
Define the dimension with the name *name* and the length *len* in the dataset *ncid*. The id of the dimension is returned.

3.3.6 netcdf.defGrp

new_ncid = netcdf.defGrp(*ncid*,*name*)
Define a group in a NetCDF file.

See also: netcdf.inqGrps.

3.3.7 netcdf.defVar

varid = netcdf.defVar(*ncid*,*name*,*xtype*,*dimids*)
Defines a variable with the name *name* in the dataset *ncid*. *xtype* can be "byte", "ubyte", "short", "ushort", "int", "uint", "int64", "uint64", "float", "double", "char" or the corresponding number as returned by netcdf.getConstant. The parameter *dimids* define the ids of the dimension. For scalar this parameter is the empty array ([]). The variable id is returned.

3.3.8 netcdf.defVarChunking

netcdf.defVarChunking (*ncid*,*varid*,*storage*,*chunkSizes*)
Define the chunking settings of NetCDF variable *varid*. If *storage* is the string "chunked", the variable is stored by chunk of the size *chunkSizes*. If *storage* is the string "contiguous", the variable is stored in a contiguous way.

3.3.9 netcdf.defVarDeflate

netcdf.defVarDeflate (*ncid*,*varid*,*shuffle*,*deflate*,*deflate_level*)
Define the compression settings NetCDF variable *varid*. If *deflate* is true, then the variable is compressed. The compression level *deflate_level* is an integer between 0 (no compression) and 9 (maximum compression).

3.3.10 netcdf.defVarFill

netcdf.defVarFill(*ncid*,*varid*,*no_fill*,*fillvalue*)
Define the fill-value settings of the NetCDF variable *varid*. If *no_fill* is false, then the values between no-contiguous writes are filled with the value *fill_value*. This is disabled by setting *no_fill* to true.

3.3.11 netcdf.defVarFletcher32

netcdf.defVarFletcher32(*ncid*,*varid*,*checksum*)
Defines the checksum settings of the variable with the id *varid* in the data set *ncid*. If *checksum* is the string "FLETCHER32", then fletcher32 checksums will be turned on for this variable. If *checksum* is "NOCHECKSUM", then checksums will be disabled.

3.3.12 netcdf.defVlen

varid = netcdf.defVlen(*ncid*,*typename*,*basetype*)
Defines a NC_VLEN variable length array type with the type name *typename* and a base datatype of *basetype* in the dataset *ncid*. *basetype* can be "byte", "ubyte", "short", "ushort", "int", "uint", "int64", "uint64", "float", "double", "char" or the corresponding number as returned by netcdf.getConstant. The new data type id is returned.

3.3.13 netcdf.delAtt

netcdf.delAtt(*ncid*,*varid*,*name*)
Deletes the attribute named *name* of the variable *varid* in the data set *ncid*. To delete a global attribute use netcdf.getConstant("global") for *varid*.

See also: netcdf.defAtt,netcdf.getConstant.

3.3.14 netcdf.endDef

netcdf.endDef (*ncid*)
Leaves define-mode of NetCDF file *ncid*.

3.3.15 netcdf.getAtt

data = netcdf.getAtt (*ncid*,*varid*,*name*)
Get the value of a NetCDF attribute. This function returns the value of the attribute called *name* of the variable *varid* in the NetCDF file *ncid*. For global attributes *varid* can be netcdf.getConstant("global").

See also: netcdf.putAtt.

3.3.16 netcdf.getChunkCache

[*size*, *nelems*, *preemption*] = netcdf.getChunkCache()
Gets the default chunk cache settings in the HDF5 library.

3.3.17 netcdf.getConstant

value = netcdf.getConstant(*name*)
Returns the value of a NetCDF constant called *name*.

See also: netcdf.getConstantNames.

3.3.18 netcdf.getConstantNames

value = netcdf.getConstantNames()
Returns a list of all constant names.

3.3.19 netcdf.getVar

data = netcdf.getVar (*ncid*,*varid*)
data = netcdf.getVar (*ncid*,*varid*,*start*)
data = netcdf.getVar (*ncid*,*varid*,*start*,*count*)
data = netcdf.getVar (*ncid*,*varid*,*start*,*count*,*stride*)
Get the data from a NetCDF variable. The data *data* is loaded from the variable *varid* of the NetCDF file *ncid*. *start* is the start index of each dimension (0-based and defaults to a vector of zeros), *count* is the number of elements of to be written along each dimension (default all elements) and *stride* is the sampling interval.

3.3.20 netcdf.inq

`[ndims,nvars,ngatts,unlimdimid] = netcdf.inq(ncid)`
Return the number of dimension (*ndims*), the number of variables (*nvars*), the number of global attributes (*ngatts*) and the id of the unlimited dimension (*unlimdimid*). If no unlimited dimension is declared -1 is returned. For NetCDF4 files, one should use the function netcdf.inqUnlimDims as multiple unlimite dimension exists.

3.3.21 netcdf.inqAtt

`[xtype,len] = netcdf.inqAtt(ncid,varid,name)`
Get attribute type and length.

See also: netcdf.inqAttName.

3.3.22 netcdf.inqAttID

`attnum = netcdf.inqAttID(ncid,varid,attname)`
Return the attribute id *attnum* of the attribute named *attname* of the variable *varid* in the dataset *ncid*. For global attributes *varid* can be netcdf.getConstant("global").

See also: netcdf.inqAttName.

3.3.23 netcdf.inqAttName

`name = netcdf.inqAttName (ncid,varid,attnum)`
Get the name of a NetCDF attribute. This function returns the name of the attribute with the id *attnum* of the variable *varid* in the NetCDF file *ncid*. For global attributes *varid* can be netcdf.getConstant("global").

See also: netcdf.inqAttName.

3.3.24 netcdf.inqDim

`[name,length] = netcdf.inqDim(ncid,dimid)`
Returns the name and length of a NetCDF dimension.

See also: netcdf.inqDimID.

3.3.25 netcdf.inqDimID

`dimid = netcdf.inqDimID(ncid,dimname)`
Return the id of a NetCDF dimension.

See also: netcdf.inqDim.

3.3.26 netcdf.inqDimIDs

`dimids = netcdf.inqDimID(ncid)`
`dimids = netcdf.inqDimID(ncid,include_parents)`
Return the dimension ids defined in a NetCDF file. If *include_parents* is 1, the dimension ids of the parent group are also returned. Per default this is not the case (*include_parents* is 0).

See also: netcdf.inqDim.

3.3.27 netcdf.inqFormat

`format = netcdf.inqFormat(ncid)`
Return the NetCDF format of the dataset *ncid*. Format might be one of the following "FORMAT_CLASSIC", "FORMAT_64BIT", "FORMAT_NETCDF4" or "FORMAT_NETCDF4_CLASSIC"

3.3.28 netcdf.inqGrpFullNcid

grp_ncid = netcdf.inqGrpFullNcid(*ncid*,*name*)
Return the group id based on the full group name.

See also: netcdf.inqGrpName.

3.3.29 netcdf.inqGrpName

name = netcdf.inqGrpName(*ncid*)
Return group name in a NetCDF file.

See also: netcdf.inqGrps.

3.3.30 netcdf.inqGrpNameFull

name = netcdf.inqGrpNameFull(*ncid*)
Return full name of group in NetCDF file.

See also: netcdf.inqGrpName.

3.3.31 netcdf.inqGrpParent

parent_ncid = netcdf.inqGrpParent(*ncid*)
Return id of the parent group

See also: netcdf.inqGrpName.

3.3.32 netcdf.inqGrps

ncids = netcdf.inqGrps(*ncid*)
Return all groups ids in a NetCDF file.

See also: netcdf.inqGrps.

3.3.33 netcdf.inqLibVers

vers = netcdf.inqLibVers()
Returns the version of the NetCDF library.

3.3.34 netcdf.inqNcid

grp_ncid = netcdf.inqNcid(*ncid*,*name*)
Return group id based on its name

See also: netcdf.inqGrpFullNcid.

3.3.35 netcdf.inqUnlimDims

unlimdimids = netcdf.inqUnlimDims(*ncid*)
Return the id of all unlimited dimensions of the NetCDF file *ncid*.

3.3.36 netcdf.inqUserType

[*typename*, *bytesize*, *basetypeid*, *numfields*, *classid*] =
netcdf.inqUserType(*ncid*,*typeid*)
Provide information on a user defined type *typeid* in the dataset *ncid*.

The function returns the typename, bytesize, base type id, number of fields and class identifier of the type.

3.3.37 netcdf.inqVar

`[name,nctype,dimids,nattr] = netcdf.inqVar (ncid,varid)`
Inquires information about a NetCDF variable. This functions returns the *name*, the NetCDF type *nctype*, an array of dimension ids *dimids* and the number of attributes *nattr* of the NetCDF variable. *nctype* in an integer corresponding NetCDF constants.

See also: netcdf.inqVarID,netcdf.getConstant.

3.3.38 netcdf.inqVarChunking

`[storage,chunkSizes] = netcdf.inqVarChunking (ncid,varid)`
Determines the chunking settings of NetCDF variable *varid*. If *storage* is the string "chunked", the variable is stored by chunk of the size *chunkSizes*. If *storage* is the string "contiguous", the variable is stored in a contiguous way.

3.3.39 netcdf.inqVarDeflate

`[shuffle,deflate,deflate_level] = netcdf.inqVarDeflate (ncid,varid)`
Determines the compression settings NetCDF variable *varid*. If *deflate* is true, then the variable is compressed. The compression level *deflate_level* is an integer between 0 (no compression) and 9 (maximum compression).

3.3.40 netcdf.inqVarFill

`[no_fill,fillvalue] = netcdf.inqVarFill(ncid,varid)`
Determines the fill-value settings of the NetCDF variable *varid*. If *no_fill* is false, then the values between no-contiguous writes are filled with the value *fill_value*. This is disabled by setting *no_fill* to true.

3.3.41 netcdf.inqVarFletcher32

`checksum = netcdf.inqVarFletcher32(ncid,varid)`
Determines the checksum settings of the variable with the id *varid* in the data set *ncid*. If fletcher32 checksums is turned on for this variable, then *checksum* is the string "FLETCHER32". Otherwise it is the string "NOCHECKSUM".

3.3.42 netcdf.inqVarID

`varid = netcdf.inqVarID (ncid,name)`
Return the id of a variable based on its name.

See also: netcdf.defVar,netcdf.inqVarIDs.

3.3.43 netcdf.inqVarIDs

`varids = netcdf.inqVarID (ncid)`
Return all variable ids. This functions returns all variable ids in a NetCDF file or NetCDF group.

See also: netcdf.inqVarID.

3.3.44 netcdf.inqVlen

`[typename, bytesize, basetypeid] = netcdf.inqVlen(ncid,typeid)`
Provide information on a NC_VLEN variable length array type *typeid* in the dataset *ncid*.

The function returns the typename, bytesize, and base type id.

3.3.45 netcdf.open

`ncid = netcdf.open(filename,mode)`
Opens the file named *filename* in the mode *mode*.

3.3.46 netcdf.putAtt

`netcdf.putAtt (ncid,varid,name,data)`
Defines a NetCDF attribute. This function defines the attribute called *name* of the variable *varid* in the NetCDF file *ncid*. The value of the attribute will be *data*. For global attributes *varid* can be netcdf.getConstant("global").

See also: netcdf.getAtt.

3.3.47 netcdf.putVar

`netcdf.putVar (ncid,varid,data)`
`netcdf.putVar (ncid,varid,start,data)`
`netcdf.putVar (ncid,varid,start,count,data)`
`netcdf.putVar (ncid,varid,start,count,stride,data)`
Put data in a NetCDF variable. The data *data* is stored in the variable *varid* of the NetCDF file *ncid*. *start* is the start index of each dimension (0-based and defaults to a vector of zeros), *count* is the number of elements of to be written along each dimension (default all elements) and *stride* is the sampling interval.

3.3.48 netcdf.reDef

`netcdf.reDef (ncid)`
Enter define-mode of NetCDF file *ncid*.

3.3.49 netcdf.renameAtt

`netcdf.renameAtt(ncid,varid,old_name,new_name)`
Renames the attribute named *old_name* of the variable *varid* in the data set *ncid*. *new_name* is the new name of the attribute. To rename a global attribute use netcdf.getConstant("global") for *varid*.

See also: netcdf.copyAtt,netcdf.getConstant.

3.3.50 netcdf.renameDim

`netcdf.renameDim(ncid,dimid,name)`
Renames the dimension with the id *dimid* in the data set *ncid*. *name* is the new name of the dimension.

3.3.51 netcdf.renameVar

`netcdf.renameVar(ncid,varid,name)`
Renames the variable with the id *varid* in the data set *ncid*. *name* is the new name of the variable.

3.3.52 netcdf.setChunkCache

`netcdf.setChunkCache(size, nelems, preemption)`
Sets the default chunk cache settings in the HDF5 library. The settings applies to all files which are subsequently opened or created.

3.3.53 netcdf.setDefaultFormat

`old_format = netcdf.setDefaultFormat(format)`
Sets the default format of the NetCDF library and returns the previous default format (as a numeric value). *format* can be "format_classic", "format_64bit", "format_netcdf4" or "format_netcdf4_classic".

3.3.54 netcdf.setFill

`old_mode = netcdf.setFill(ncid,fillmode)`
Change the fill mode (*fillmode*) of the data set *ncid*. The previous value of the fill mode is returned. *fillmode* can be either "fill" or "nofill".

3.3.55 netcdf.sync

`netcdf.sync(ncid)`
Writes all changes to the disk and leaves the file open.

3.4 Import functions (Deprecated)

3.4.1 import_netcdf

`import_fits`
Dummy function provided to provide compatibility with older versions of GNU Octave netcdf Function is deprecated.

3.5 Test function

3.5.1 test_netcdf

`test_netcdf`
Function to do a basic test of the netcdf interface

Appendix A GNU General Public License

Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program—to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

   "This License" refers to version 3 of the GNU General Public License.

   "Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

   "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

   To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

   A "covered work" means either the unmodified Program or a work based on the Program.

   To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

   To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

   An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

   The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

   A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

   The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

   The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

   All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

   You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

   Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

   When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

   You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

   You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

   a. The work must carry prominent notices stating that you modified it, and giving a relevant date.
   b. The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
   c. You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable

section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d. If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c. Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d. Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e. Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a

consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a. Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b. Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c. Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d. Limiting the use for publicity purposes of names of licensors or authors of the material; or

e. Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f. Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

    Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

    An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

    You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

    A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

    A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

    Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

    In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

    If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

    If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

    If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

    Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

    The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

    If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

    Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

    THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS"

WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

    If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and a brief idea of what it does.
Copyright (C) year name of author

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or (at
your option) any later version.

This program is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see http://www.gnu.org/licenses/.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
program Copyright (C) year name of author
This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
```

```
This is free software, and you are welcome to redistribute it
under certain conditions; type ‘show c’ for details.
```

The hypothetical commands ‘`show w`’ and ‘`show c`’ should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an “about box”.

You should also get your employer (if you work as a programmer) or school, if any, to sign a “copyright disclaimer” for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see http://www.gnu.org/licenses/.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read http://www.gnu.org/philosophy/why-not-lgpl.html.

Index

O

P

R

S

T

W