Package ‘rcdo’

November 28, 2025

Title Wrapper of 'CDO' Operators

Version 0.3.2

Description Provides a translation layer between 'R' and 'CDO' operators. Each operator is it's own function with documentation. Nested or piped functions will be translated into 'CDO' chains.

License GPL (>= 3)

Encoding UTF-8

Language en-GB

RoxygenNote 7.3.3

Depends R (>= 3.5.0)

SystemRequirements cdo

URL `https://eliocamp.github.io/rcdo/`, `https://github.com/eliocamp/rcdo`

Imports cli, R6, rlang

BugReports `https://github.com/eliocamp/rcdo/issues`

Suggests glue, knitr, markdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Elio Campitelli [cre, aut, cph] (ORCID: <`https://orcid.org/0000-0002-7742-9230`>), MPI für Meteorologie [cph]

Maintainer Elio Campitelli <`eliocampitelli@gmail.com`>

Repository CRAN

Date/Publication 2025-11-28 13:00:02 UTC

Contents

`adisit` *Potential temperature to insitu temperature and vice versa*

Description

Potential temperature to insitu temperature and vice versa

Usage

```
cdo_adipot(ifile, pressure = NULL, ofile = NULL)

cdo_adisit(ifile, pressure = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`pressure`	FLOAT - Pressure in bar (constant value assigned to all levels)
`ofile`	String with the path to the output file.

Details

```
adisit  Potential temperature to in-situ temperature
     This is a special operator for the post processing of the ocean and sea ice model MPIOM.
     It converts potential temperature adiabatically to in-situ temperature to(t, s, p).
     Required input fields are sea water potential temperature (name=tho; code=2) and sea water salinit
     Pressure is calculated from the level information or can be specified by the optional parameter.
     Output fields are sea water temperature (name=to; code=20) and sea water salinity (name=s; code=5)
adipot  In-situ temperature to potential temperature
     This is a special operator for the post processing of the ocean and sea ice model MPIOM.
     It converts in-situ temperature to potential temperature tho(to, s, p). Required input fields
     are sea water in-situ temperature (name=t; code=2) and sea water salinity (name=sao,s; code=5).
     Pressure is calculated from the level information or can be specified by the optional parameter.
     Output fields are sea water temperature (name=tho; code=2) and sea water salinity (name=s; code=5)
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`afterburner` *ECHAM standard post processor*

Description

The "afterburner" is the standard post processor for ECHAM GRIB and NetCDF data which provides the following operations: - Extract specified variables and levels - Compute derived variables - Transform spectral data to Gaussian grid representation - Vertical interpolation to pressure levels - Compute temporal means This operator reads selection parameters as namelist from stdin. Use the UNIX redirection "<namelistfile" to read the namelist from file. The input files can't be combined with other CDO operators because of an optimized reader for this operator.

Usage

```
cdo_after(ifiles, vct = NULL, ofile = NULL)
```

Arguments

`ifiles`	Character vector with the path to the input files.
`vct`	STRING - File with VCT in ASCII format
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`apply` *Apply operators*

Description

The apply utility runs the named operators on each input file. The input files must be enclosed in square brackets. This utility can only be used on a series of input files. These are all operators with more than one input file (infiles). Here is an incomplete list of these operators: copy, cat, merge, mergetime, select, ENSSTAT. The parameter operators is a blank-separated list of CDO operators. Use quotation marks if more than one operator is needed. Each operator may have only one input and output stream.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`arith` *Arithmetic on two datasets*

Description

This module performs simple arithmetic of two datasets. The number of fields in infile1 should be the same as in infile2. The fields in outfile inherit the meta data from infile1. All operators in this module simply process one field after the other from the two input files. Neither the order of the variables nor the date is checked. One of the input files can contain only one timestep or one variable.

Usage

```
cdo_add(ifile1, ifile2, ofile = NULL)

cdo_atan2(ifile1, ifile2, ofile = NULL)

cdo_div(ifile1, ifile2, ofile = NULL)

cdo_max(ifile1, ifile2, ofile = NULL)
```

```
cdo_min(ifile1, ifile2, ofile = NULL)

cdo_mul(ifile1, ifile2, ofile = NULL)

cdo_sub(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Details

```
add    Add two fields
       o(t,x) = i_1(t,x) + i_2(t,x)
sub    Subtract two fields
       o(t,x) = i_1(t,x) - i_2(t,x)
mul    Multiply two fields
       o(t,x) = i_1(t,x) * i_2(t,x)
div    Divide two fields
       o(t,x) = i_1(t,x) / i_2(t,x)
min    Minimum of two fields
       o(t,x) = min(i_1(t,x), i_2(t,x))
max    Maximum of two fields
       o(t,x) = max(i_1(t,x), i_2(t,x))
atan2  Arc tangent of two fields
      The atan2 operator calculates the arc tangent of two fields. The result is
       in radians, which is between -PI and PI (inclusive).

       o(t,x) = atan2(i_1(t,x), i_2(t,x))
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`arithc` *Arithmetic with a constant*

Description

This module performs simple arithmetic with all field elements of a dataset and a constant. The fields in outfile inherit the meta data from infile.

Usage

```
cdo_addc(ifile, c = NULL, ofile = NULL)

cdo_divc(ifile, c = NULL, ofile = NULL)

cdo_maxc(ifile, c = NULL, ofile = NULL)

cdo_minc(ifile, c = NULL, ofile = NULL)

cdo_mulc(ifile, c = NULL, ofile = NULL)

cdo_subc(ifile, c = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`c`	FLOAT - Constant
`ofile`	String with the path to the output file.

Details

```
addc  Add a constant
      o(t,x) = i(t,x) + c
subc  Subtract a constant
      o(t,x) = i(t,x) - c
mulc  Multiply with a constant
      o(t,x) = i(t,x) * c
divc  Divide by a constant
      o(t,x) = i(t,x) / c
minc  Minimum of a field and a constant
      o(t,x) = min(i(t,x), c)
maxc  Maximum of a field and a constant
      o(t,x) = max(i(t,x), c)
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don’t return filenames return a character vector with the string output.

arithdays *Arithmetic with days*

Description

This module multiplies or divides each timestep of a dataset with the corresponding days per month or days per year. The result of these functions depends on the used calendar of the input data.

Usage

```
cdo_divdpm(ifile, ofile = NULL)

cdo_divdpy(ifile, ofile = NULL)

cdo_muldpm(ifile, ofile = NULL)

cdo_muldpy(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Details

```
muldpm  Multiply with days per month
        o(t,x) = i(t,x) * days_per_month
divdpm  Divide by days per month
        o(t,x) = i(t,x) / days_per_month
muldpy  Multiply with days per year
        o(t,x) = i(t,x) * days_per_year
divdpy  Divide by days per year
        o(t,x) = i(t,x) / days_per_year
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`arithlat` *Arithmetic with latitude*

Description

This module multiplies or divides each field element with the cosine of the latitude.

Usage

```
cdo_divcoslat(ifile, ofile = NULL)

cdo_mulcoslat(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Details

```
mulcoslat  Multiply with the cosine of the latitude
           o(t,x) = i(t,x) * cos(latitude(x))
divcoslat  Divide by cosine of the latitude
           o(t,x) = i(t,x) / cos(latitude(x))
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`bitrounding` *Bit rounding*

Description

This operator calculates for each field the number of necessary mantissa bits to get a certain information level in the data. With this number of significant bits (numbits) a rounding of the data is performed. This allows the data to be compressed to a higher level. The default value of the information level is 0.9999 and can be adjusted with the parameter inflevel. That means 99.99% of the information in the mantissa bits is preserved. Alternatively, the number of significant bits can be set for all variables with the numbits parameter. Furthermore, numbits can be assigned for each variable via the filename parameter. In this case, numbits is still calculated for all variables if they are not present in the file. The analysis of the bit information is based on the Julia library BitInformation.jl (https://github.com/milankl/BitInformation.jl). The procedure to derive the number of significant mantissa bits was adapted from the Python library xbitinfo (https://github.com/observingClouds/xbitinfo). Quantize to the number of mantissa bits is done with IEEE rounding using code from NetCDF 4.9.0. Currently only 32-bit float data is rounded. Data with missing values are not yet supported for the calculation of significant bits.

Usage

```
cdo_bitrounding(
  ifile,
  inflevel = NULL,
  addbits = NULL,
  minbits = NULL,
  maxbits = NULL,
  numsteps = NULL,
  numbits = NULL,
  printbits = NULL,
  filename = NULL,
  ofile = NULL
)
```

Arguments

`ifile`	String with the path to the input file.
`inflevel`	FLOAT - Information level (0 - 1) [default: 0.9999]
`addbits`	INTEGER - Add bits to the number of significant bits [default: 0]
`minbits`	INTEGER - Minimum value of the number of bits [default: 1]
`maxbits`	INTEGER - Maximum value of the number of bits [default: 23]
`numsteps`	INTEGER - Set to 1 to run the calculation only in the first time step
`numbits`	INTEGER - Set number of significant bits
`printbits`	BOOL - Print max. numbits per variable of 1st timestep to stdout [format: name=numbits]
`filename`	STRING - Read number of significant bits per variable from file [format: name=numbits]
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

cdo *Execute a custom operator*

Description

Execute a custom operator

Usage

```
cdo(operator, input, params = NULL, output = NULL)

cdo_operator(command, params, n_input, n_output)
```

Arguments

`operator`	a list created with `cdo_operator`.
`input`	a list with the input files.
`params`	a character vector with the name of the parameter
`output`	a vector of file name(s).
`command`	a string with the command used to run the operator
`n_input, n_output`	an integer with the number of input and output files required by the operator

Value

a cdo operation.

A list with elements command, params, n_input and n_output.

cdo_cache_set *Manages the cache*

Description

Manages whether cdo will try to recover existing files if available.

Usage

```
cdo_cache_set(cache = tempdir())

cdo_cache_get()

cdo_cache_unset()
```

Arguments

`cache`	either the location of the default cache or a list which is the result of a previous `cdo_cache_set()` call.

Details

When first executing the operation, `cdo_execute()` will create a ".hash" file matching the output file name with a hash generated from the current cdo version, the text of the command, the sum of the file sizes of the input files and the most recent modified time of the input files. The next time the same command is executed, if the cache is active, `cdo_execute` will compute the same hash and compare it with the file and, if it matches, it will return the output file without running the command. Caching currently only works with operations with only one output file.

These functions change the global options. If used inside functions, it's generally a good idea to reset the original values before exiting the function with `on.exit()`.

Value

A list with the old values of the `rcdo_cache` and `rcdo_tmpdir` options.

Examples

```
# Set the cache
old <- cdo_cache_set(cache = "data/cache")

# Reset the cache to its previous state
cdo_cache_set(old)

# Disable the cache
old <- cdo_cache_unset()

# Again, reset the cache to its previous state.
cdo_cache_set(old)

with_cache <- function(operation, cache) {
  old <- cdo_cache_set(cache)
  on.exit(cdo_cache_set(old))

  # Rest of the function
}

without_cache <- function(operation) {
  old <- cdo_cache_unset(cache)
  on.exit(cdo_cache_set(old))

  # Rest of the function
}
```

cdo_execute *Execute a CDO operation*

Description

Execute a CDO operation

Usage

```
cdo_execute(
  operation,
  output = temp_output(operation, !cache),
  options = NULL,
  options_replace = FALSE,
  verbose = FALSE,
  cache = getOption("rcdo_cache", default = FALSE)
)

cdo_execute_list(
  operations,
  output = NULL,
  options = NULL,
  options_replace = FALSE,
  verbose = FALSE,
  cache = FALSE
)
```

Arguments

`operation`	a CDO operation
`output`	an output file or base string for output files. Defaults to temporary files that will be deleted when its bond variable is garbage collected.
`options`	character vector with CDO options.
`options_replace`	logical indicating whether the options given in execute should replace any other options (global or set with `cdo_options_use`).
`verbose`	whether to print the command being executed.
`cache`	whether to cache results. See `cdo_cache_set()` for details.
`operations`	a list of CDO operations

`cdo_install` *Install the supported CDO version*

Description

Install the supported CDO version

Usage

```
cdo_install(
  reinstall = FALSE,
  proj = "/usr",
  netcdf = "/usr",
  fftw3 = "/usr",
  eccodes = "/usr"
)
```

Arguments

`reinstall`	Logical. Set to true to force reinstallation.
`proj`, `netcdf`, `fftw3`, `eccodes`	Location of the optional libraries.

Details

rcdo should work with your normal CDO installation but you if your installed version is not the one used to generate this package, there could be some small inconsistencies in the documentation, missing operators, extra operators or changes in syntax.

`cdo_install()` will attempt to download, configure, compile and install CDO version 2.5.1 in the package data directory. If this version of CDO exists, the package will use it. Otherwise, it will use your system's installation.

Value

The path to the installed cdo executable.

`cdo_options_use` *Manage CDO options*

Description

Set the options of operations.

Usage

```
cdo_options_use(operation, options)

cdo_options_set(options)

cdo_options_get(options)

cdo_options_clear()
```

Arguments

`operation`	operation to add options to.
`options`	character vector with CDO options.

Details

`cdo_options_use()` takes an operation and adds a set of options to be used in that operation. `cdo_options_set()` sets the default options that all operations should use by default. You can retrieve the default options with `cdo_options_get()` or clear all default options with `cdo_options_clear()` or `cdo_options_set(NULL)`.

`cdo_set_output` *Set output and options*

Description

Set output and options

Usage

```
cdo_set_output(operation, output)
```

Arguments

`operation`	a CDO operation
`output`	an output file or base string for output files

`cdo_use` *Chose CDO version to use*

Description

Chose CDO version to use

Usage

```
cdo_use(version = c("system", "packaged"))
```

Arguments

`version`	String with the cdo version to use: • `"system"` (the default) will use the system-wide installed version (specifically, whatever path is returned by `Sys.which("cdo")`). • `"packaged"` instructs rcdo to use a package-specific version that can be compiled and installed with `cdo_install()`.

Details

A one-time warning will be issued if the the cdo version found when using `"system"` doesn't match the version used to build the rcdo package. In that case, some operators documented in this package might not be available to you or might behave slightly different. However, most operators are stable, particularly the most often used ones.

Value

The path to the cdo executable (invisibly).

`change` *Change field header*

Description

This module reads fields from infile, changes some header values and writes the results to outfile. The kind of changes depends on the chosen operator.

Usage

```
cdo_chcode(
  ifile,
  code = NULL,
  oldcode = NULL,
  newcode = NULL,
  oldparam = NULL,
  newparam = NULL,
  name = NULL,
  oldname = NULL,
  newname = NULL,
  oldlev = NULL,
  newlev = NULL,
  ofile = NULL
)

cdo_chlevel(
  ifile,
  code = NULL,
  oldcode = NULL,
  newcode = NULL,
  oldparam = NULL,
  newparam = NULL,
  name = NULL,
  oldname = NULL,
  newname = NULL,
  oldlev = NULL,
  newlev = NULL,
  ofile = NULL
)

cdo_chlevelc(
  ifile,
  code = NULL,
  oldcode = NULL,
  newcode = NULL,
  oldparam = NULL,
  newparam = NULL,
  name = NULL,
  oldname = NULL,
  newname = NULL,
  oldlev = NULL,
  newlev = NULL,
  ofile = NULL
)

cdo_chlevelv(
  ifile,
  code = NULL,
  oldcode = NULL,
  newcode = NULL,
  oldparam = NULL,
  newparam = NULL,
  name = NULL,
  oldname = NULL,
  newname = NULL,
  oldlev = NULL,
  newlev = NULL,
  ofile = NULL
)

cdo_chname(
  ifile,
  code = NULL,
  oldcode = NULL,
  newcode = NULL,
  oldparam = NULL,
  newparam = NULL,
  name = NULL,
  oldname = NULL,
  newname = NULL,
  oldlev = NULL,
  newlev = NULL,
  ofile = NULL
)

cdo_chparam(
  ifile,
  code = NULL,
  oldcode = NULL,
  newcode = NULL,
  oldparam = NULL,
  newparam = NULL,
  name = NULL,
  oldname = NULL,
  newname = NULL,
  oldlev = NULL,
  newlev = NULL,
  ofile = NULL
)

cdo_chunit(
  ifile,
  code = NULL,
  oldcode = NULL,
  newcode = NULL,
  oldparam = NULL,
  newparam = NULL,
  name = NULL,
  oldname = NULL,
  newname = NULL,
  oldlev = NULL,
  newlev = NULL,
  ofile = NULL
)
```

Arguments

`ifile`	String with the path to the input file.
`code`	INTEGER - Code number
`oldcode`	INTEGER - Pairs of old and new code numbers
`newcode`	INTEGER - Pairs of old and new code numbers
`oldparam`	STRING - Pairs of old and new parameter identifiers
`newparam`	STRING - Pairs of old and new parameter identifiers
`name`	STRING - Variable name
`oldname`	STRING - Pairs of old and new variable names
`newname`	STRING - Pairs of old and new variable names
`oldlev`	FLOAT - Old level
`newlev`	FLOAT - New level
`ofile`	String with the path to the output file.

Details

```
chcode    Change code number
          Changes some user given code numbers to new user given values.
chparam   Change parameter identifier
         Changes some user given parameter identifiers to new user given values.
chname    Change variable or coordinate name
      Changes some user given variable or coordinate names to new user given names.
chunit    Change variable unit
          Changes some user given variable units to new user given units.
chlevel   Change level
          Changes some user given levels to new user given values.
chlevelc  Change level of one code
          Changes one level of a user given code number.
chlevelv  Change level of one variable
          Changes one level of a user given variable name.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`cmorlite` *CMOR lite*

Description

The CMOR (Climate Model Output Rewriter) library comprises a set of functions, that can be used to produce CF-compliant NetCDF files that fulfill the requirements of many of the climate community's standard model experiments. These experiments are collectively referred to as MIP's. Much of the metadata written to the output files is defined in MIP-specific tables, typically made available from each MIP's web site. The CDO operator cmorlite process the header and variable section of such MIP tables and writes the result with the internal IO library CDI. In addition to the CMOR 2 and 3 table format, the CDO parameter table format is also supported. The following parameter table entries are available: Entry & Type & Description name & WORD & Name of the variable out_name & WORD & New name of the variable type & WORD & Data type (real or double) standard_name & WORD & As defined in the CF standard name table long_name & STRING & Describing the variable units & STRING & Specifying the units for the variable comment & STRING & Information concerning the variable cell_methods & STRING & Information concerning calculation of means or climatologies cell_measures & STRING & Indicates the names of the variables containing cell areas and volumes missing_value & FLOAT & Specifying how missing data will be identified valid_min & FLOAT & Minimum valid value valid_max & FLOAT & Maximum valid value ok_min_mean_abs & FLOAT & Minimum absolute mean ok_max_mean_abs & FLOAT & Maximum absolute mean factor & FLOAT & Scale factor delete & INTEGER & Set to 1 to delete variable convert & INTEGER & Set to 1 to convert the unit if necessary Most of the above entries are stored as variables attributes, some of them are handled differently. The variable name is used as a search key for the parameter table. valid_min, valid_max, ok_min_mean_abs and ok_max_mean_abs are used to check the range of the data.

Usage

```
cdo_cmorlite(ifile, table = NULL, convert = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`table`	STRING - Name of the CMOR table as specified from PCMDI
`convert`	STRING - Converts the units if necessary
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`collgrid` *Collect horizontal grid*

Description

This operator collects the data of the input files to one output file. All input files need to have the same variables and the same number of timesteps on a different horizonal grid region. If the source regions are on a structured lon/lat grid, all regions together must result in a new structured lat/long grid box. Data on an unstructured grid is concatenated in the order of the input files. The parameter nx needs to be specified only for curvilinear grids.

Usage

```
cdo_collgrid(ifiles, nx = NULL, names = NULL, ofile = NULL)
```

Arguments

`ifiles`	Character vector with the path to the input files.
`nx`	INTEGER - Number of regions in x direction [default: number of input files]
`names`	STRING - Comma-separated list of variable names [default: all variables]
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

Note

This operator needs to open all input files simultaneously. The maximum number of open files depends on the operating system!

`comp` *Comparison of two fields*

Description

This module compares two datasets field by field. The resulting field is a mask containing 1 if the comparison is true and 0 if not. The number of fields in infile1 should be the same as in infile2. One of the input files can contain only one timestep or one field. The fields in outfile inherit the meta data from infile1 or infile2. The type of comparison depends on the chosen operator.

Usage

```
cdo_eq(ifile1, ifile2, ofile = NULL)

cdo_ge(ifile1, ifile2, ofile = NULL)

cdo_gt(ifile1, ifile2, ofile = NULL)

cdo_le(ifile1, ifile2, ofile = NULL)

cdo_lt(ifile1, ifile2, ofile = NULL)

cdo_ne(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1`, `ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Details

```
eq  Equal
              /   1   if i_1(t,x) EQ i_2(t,x)  AND  i_1(t,x),i_2(t,x) NE miss
   o(t,x) = &lt;    0   if i_1(t,x) NE i_2(t,x)  AND  i_1(t,x),i_2(t,x) NE miss
              \\  miss if i_1(t,x) EQ miss      OR   i_2(t,x) EQ miss
ne  Not equal
              /   1   if i_1(t,x) NE i_2(t,x)  AND  i_1(t,x),i_2(t,x) NE miss
   o(t,x) = &lt;    0   if i_1(t,x) EQ i_2(t,x)  AND  i_1(t,x),i_2(t,x) NE miss
              \\  miss if i_1(t,x) EQ miss      OR   i_2(t,x) EQ miss
le  Less equal
              /   1   if i_1(t,x) LE i_2(t,x)  AND  i_1(t,x),i_2(t,x) NE miss
   o(t,x) = &lt;    0   if i_1(t,x) GT i_2(t,x)  AND  i_1(t,x),i_2(t,x) NE miss
              \\  miss if i_1(t,x) EQ miss      OR   i_2(t,x) EQ miss
lt  Less than
              /   1   if i_1(t,x) LT i_2(t,x)  AND  i_1(t,x),i_2(t,x) NE miss
   o(t,x) = &lt;    0   if i_1(t,x) GE i_2(t,x)  AND  i_1(t,x),i_2(t,x) NE miss
              \\  miss if i_1(t,x) EQ miss      OR   i_2(t,x) EQ miss
ge  Greater equal
              /   1   if i_1(t,x) GE i_2(t,x)  AND  i_1(t,x),i_2(t,x) NE miss
   o(t,x) = &lt;    0   if i_1(t,x) LT i_2(t,x)  AND  i_1(t,x),i_2(t,x) NE miss
              \\  miss if i_1(t,x) EQ miss      OR   i_2(t,x) EQ miss
gt  Greater than
              /   1   if i_1(t,x) GT i_2(t,x)  AND  i_1(t,x),i_2(t,x) NE miss
   o(t,x) = &lt;    0   if i_1(t,x) LE i_2(t,x)  AND  i_1(t,x),i_2(t,x) NE miss
              \\  miss if i_1(t,x) EQ miss      OR   i_2(t,x) EQ miss
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

compc *Comparison of a field with a constant*

Description

This module compares all fields of a dataset with a constant. The resulting field is a mask containing 1 if the comparison is true and 0 if not. The type of comparison depends on the chosen operator.

Usage

```
cdo_eqc(ifile, c = NULL, ofile = NULL)

cdo_gec(ifile, c = NULL, ofile = NULL)

cdo_gtc(ifile, c = NULL, ofile = NULL)

cdo_lec(ifile, c = NULL, ofile = NULL)

cdo_ltc(ifile, c = NULL, ofile = NULL)

cdo_nec(ifile, c = NULL, ofile = NULL)
```

Arguments

ifile	String with the path to the input file.
c	FLOAT - Constant
ofile	String with the path to the output file.

Details

```
eqc  Equal constant
               /   1   if i(t,x) EQ c     AND  i(t,x),c NE miss
     o(t,x) = &lt;    0   if i(t,x) NE c     AND  i(t,x),c NE miss
               \\  miss if i(t,x) EQ miss  OR   c EQ miss
nec  Not equal constant
               /   1   if i(t,x) NE c     AND  i(t,x),c NE miss
     o(t,x) = &lt;    0   if i(t,x) EQ c     AND  i(t,x),c NE miss
               \\  miss if i(t,x) EQ miss  OR   c EQ miss
lec  Less equal constant
               /   1   if i(t,x) LE c     AND  i(t,x),c NE miss
     o(t,x) = &lt;    0   if i(t,x) GT c     AND  i(t,x),c NE miss
               \\  miss if i(t,x) EQ miss  OR   c EQ miss
ltc  Less than constant
```

```
              /   1   if i(t,x) LT c     AND  i(t,x),c NE miss
     o(t,x) = &lt;    0   if i(t,x) GE c     AND  i(t,x),c NE miss
              \\  miss if i(t,x) EQ miss  OR   c EQ miss
gec  Greater equal constant
              /   1   if i(t,x) GE c     AND  i(t,x),c NE miss
     o(t,x) = &lt;    0   if i(t,x) LT c     AND  i(t,x),c NE miss
              \\  miss if i(t,x) EQ miss  OR   c EQ miss
gtc  Greater than constant
              /   1   if i(t,x) GT c     AND  i(t,x),c NE miss
     o(t,x) = &lt;    0   if i(t,x) LE c     AND  i(t,x),c NE miss
              \\  miss if i(t,x) EQ miss  OR   c EQ miss
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

---

cond *Conditional select one field*

Description

This module selects field elements from infile2 with respect to infile1 and writes them to outfile. The fields in infile1 are handled as a mask. A value not equal to zero is treated as "true zero is treated as "false". The number of fields in infile1 has either to be the same as in infile2 or the same as in one timestep of infile2 or only one. The fields in outfile inherit the meta data from infile2.

Usage

```
cdo_ifnotthen(ifile1, ifile2, ofile = NULL)

cdo_ifthen(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Details

```
ifthen     If then
                    / i_2(t,x) if i_1(t,x) NE 0  AND  i_1(t,x) NE miss
           o(t,x) =
                    \\ miss     if i_1(t,x) EQ 0  OR   i_1(t,x) EQ miss
ifnotthen  If not then
                    / i_2(t,x) if i_1(t,x) EQ 0  AND  i_1(t,x) NE miss
```

```
            o(t,x) =
                  \\ miss     if i_1(t,x) NE 0  OR   i_1(t,x) EQ miss
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`cond2` *Conditional select two fields*

Description

This operator selects field elements from infile2 or infile3 with respect to infile1 and writes them to outfile. The fields in infile1 are handled as a mask. A value not equal to zero is treated as "true zero is treated as "false". The number of fields in infile1 has either to be the same as in infile2 or the same as in one timestep of infile2 or only one. infile2 and infile3 need to have the same number of fields. The fields in outfile inherit the meta data from infile2. / i_2(t,x) if i_1(t,x) NE 0 AND i_1(t,x) NE miss o(t,x) = < i_3(t,x) if i_1(t,x) EQ 0 AND i_1[t,x) NE miss miss if i_1(t,x) EQ miss

Usage

```
cdo_ifthenelse(ifile1, ifile2, ifile3, ofile = NULL)
```

Arguments

`ifile1`, `ifile2`, `ifile3`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

condc	*Conditional select a constant*

Description

This module creates fields with a constant value or missing value. The fields in infile are handled as a mask. A value not equal to zero is treated as "true zero is treated as "false".

Usage

```
cdo_ifnotthenc(ifile, c = NULL, ofile = NULL)

cdo_ifthenc(ifile, c = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`c`	FLOAT - Constant
`ofile`	String with the path to the output file.

Details

```
ifthenc     If then constant
                     / c      if i(t,x) NE 0  AND  i(t,x) NE miss
            o(t,x) =
                     \\ miss   if i(t,x) EQ 0  OR   i(t,x) EQ miss
ifnotthenc  If not then constant
                     / c      if i(t,x) EQ 0  AND  i(t,x) NE miss
            o(t,x) =
                     \\ miss   if i(t,x) NE 0  OR   i(t,x) EQ miss
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`consecstat` *Consecute timestep periods*

Description

This module computes periods over all timesteps in infile where a certain property is valid. The property can be chosen by creating a mask from the original data, which is the expected input format for operators of this module. Depending on the operator full information about each period or just its length and ending date are computed.

Usage

```
cdo_consecsum(ifile, ofile = NULL)

cdo_consects(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Details

```
consecsum  Consecutive Sum
           This operator computes periods of consecutive timesteps similar to a
           runsum, but periods are finished, when the mask value is 0. That way
        multiple periods can be found. Timesteps from the input are preserved. Missing
          values are handled like 0, i.e. finish periods of consecutive timesteps.
consects   Consecutive Timesteps
        In contrast to the operator above consects only computes the length of each
         period together with its last timestep. To be able to perform statistical
          analysis like min, max or mean, everything else is set to missing value.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`copy` *Copy datasets*

Description

This module contains operators to copy, clone or concatenate datasets. infiles is an arbitrary number of input files. All input files need to have the same structure with the same variables on different timesteps.

Usage

```
cdo_cat(ifiles, ofile = NULL)

cdo_clone(ifiles, ofile = NULL)

cdo_copy(ifiles, ofile = NULL)
```

Arguments

`ifiles`	Character vector with the path to the input files.
`ofile`	String with the path to the output file.

Details

```
copy   Copy datasets
       Copies all input datasets to outfile.
clone  Clone datasets
     Copies all input datasets to outfile. In contrast to the copy operator, clone tries
     not to change the input data. GRIB records are neither decoded nor decompressed.
cat    Concatenate datasets
       Concatenates all input datasets and appends the result to the end
       of outfile. If outfile does not exist it will be created.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don’t return filenames return a character vector with the string output.

`dayarith` *Daily arithmetic*

Description

This module performs simple arithmetic of a time series and one timestep with the same day, month and year. For each field in infile1 the corresponding field of the timestep in infile2 with the same day, month and year is used. The input files need to have the same structure with the same variables. Usually infile2 is generated by an operator of the module DAYSTAT.

Usage

```
cdo_dayadd(ifile1, ifile2, ofile = NULL)

cdo_daydiv(ifile1, ifile2, ofile = NULL)

cdo_daymul(ifile1, ifile2, ofile = NULL)

cdo_daysub(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Details

```
dayadd  Add daily time series
        Adds a time series and a daily time series.
daysub  Subtract daily time series
        Subtracts a time series and a daily time series.
daymul  Multiply daily time series
        Multiplies a time series and a daily time series.
daydiv  Divide daily time series
        Divides a time series and a daily time series.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`daypctl` *Daily percentile values*

Description

This operator computes percentiles over all timesteps of the same day in infile1. The algorithm uses histograms with minimum and maximum bounds given in infile2 and infile3, respectively. The default number of histogram bins is 101. The default can be overridden by defining the environment variable CDO_PCTL_NBINS. The files infile2 and infile3 should be the result of corresponding daymin and daymax operations, respectively. The time of outfile is determined by the time in the middle of all contributing timesteps of infile1. This can be change with the CDO option –timestat_date <first|middle|last>. For every adjacent sequence t_1, ...,t_n of timesteps of the same day it is: o(t,x) = pth percentile {i(t',x), t_1<t'<=t_n}

Usage

```
cdo_daypctl(ifile1, ifile2, ifile3, p = NULL, ofile = NULL)
```

Arguments

`ifile1`, `ifile2`, `ifile3`	Strings with the path to the input files.
`p`	FLOAT - Percentile number in {0, ..., 100}
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`daystat` *Daily statistics*

Description

This module computes statistical values over timesteps of the same day. Depending on the chosen operator the minimum, maximum, range, sum, average, variance or standard deviation of timesteps of the same day is written to outfile. The time of outfile is determined by the time in the middle of all contributing timesteps of infile. This can be change with the CDO option –timestat_date <first|middle|last>.

Usage

```
cdo_dayavg(ifile, complete_only = NULL, ofile = NULL)

cdo_daymax(ifile, complete_only = NULL, ofile = NULL)

cdo_daymean(ifile, complete_only = NULL, ofile = NULL)

cdo_daymin(ifile, complete_only = NULL, ofile = NULL)

cdo_dayrange(ifile, complete_only = NULL, ofile = NULL)

cdo_daystd(ifile, complete_only = NULL, ofile = NULL)

cdo_daystd1(ifile, complete_only = NULL, ofile = NULL)

cdo_daysum(ifile, complete_only = NULL, ofile = NULL)

cdo_dayvar(ifile, complete_only = NULL, ofile = NULL)

cdo_dayvar1(ifile, complete_only = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`complete_only`	BOOL - Process the last day only if it is complete
`ofile`	String with the path to the output file.

Details

```
daymin    Daily minimum
       For every adjacent sequence t_1, ...,t_n of timesteps of the same day it is:

          o(t,x) = min\{i(t',x), t_1&lt;t'&lt;=t_n\}
daymax    Daily maximum
       For every adjacent sequence t_1, ...,t_n of timesteps of the same day it is:

          o(t,x) = max\{i(t',x), t_1&lt;t'&lt;=t_n\}
dayrange  Daily range
       For every adjacent sequence t_1, ...,t_n of timesteps of the same day it is:

          o(t,x) = range\{i(t',x), t_1&lt;t'&lt;=t_n\}
daysum    Daily sum
       For every adjacent sequence t_1, ...,t_n of timesteps of the same day it is:

          o(t,x) = sum\{i(t',x), t_1&lt;t'&lt;=t_n\}
daymean   Daily mean
       For every adjacent sequence t_1, ...,t_n of timesteps of the same day it is:
          o(t,x) = mean\{i(t',x), t_1&lt;t'&lt;=t_n\}
dayavg    Daily average
       For every adjacent sequence t_1, ...,t_n of timesteps of the same day it is:

          o(t,x) = avg\{i(t',x), t_1&lt;t'&lt;=t_n\}
daystd    Daily standard deviation
       Normalize by n. For every adjacent sequence t_1, ...,t_n of timesteps of the same day it is:

          o(t,x) = std\{i(t',x), t_1&lt;t'&lt;=t_n\}
daystd1   Daily standard deviation (n-1)
       Normalize by (n-1). For every adjacent sequence t_1, ...,t_n of timesteps of the same day it is:

          o(t,x) = std1\{i(t',x), t_1&lt;t'&lt;=t_n\}
dayvar    Daily variance
       Normalize by n. For every adjacent sequence t_1, ...,t_n of timesteps of the same day it is:

          o(t,x) = var\{i(t',x), t_1&lt;t'&lt;=t_n\}
dayvar1   Daily variance (n-1)
       Normalize by (n-1). For every adjacent sequence t_1, ...,t_n of timesteps of the same day it is:

          o(t,x) = var1\{i(t',x), t_1&lt;t'&lt;=t_n\}
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don’t return filenames return a character vector with the string output.

`deltat` *Difference between timesteps*

Description

This operator computes the difference between each timestep.

Usage

```
cdo_deltat(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`derivepar` *Derived model parameters*

Description

This module contains operators that calculate derived model parameters. These are currently the parameters sea level pressure and geopotential height. All necessary input variables are identified by their GRIB1 code number or the NetCDF CF standard name. Supported GRIB1 parameter tables are: WMO standard table number 2 and ECMWF local table number 128. CF standard name & Units & GRIB 1 code surface_air_pressure & Pa & 134 air_temperature & K & 130 specific_humidity & kg/kg & 133 surface_geopotential & m2 s-2 & 129 geopotential_height & m & 156

Usage

```
cdo_gheight(ifile, ofile = NULL)

cdo_gheight_half(ifile, ofile = NULL)

cdo_sealevelpressure(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Details

```
sealevelpressure  Sea level pressure
            This operator computes the sea level pressure (air_pressure_at_sea_level). Required input f
            are surface_air_pressure, surface_geopotential and air_temperature on full hybrid sigma pr
gheight           Geopotential height on full-levels
            This operator computes the geopotential height (geopotential_height) on model full-levels i
            Required input fields are surface_air_pressure, surface_geopotential, specific_humidity ar
            on full hybrid sigma pressure levels. Note, this procedure is an approximation, which doesn'
                 account the effects of e.g. cloud ice and water, rain and snow.
gheight_half      Geopotential height on half-levels
            This operator computes the geopotential height (geopotential_height) on model half-levels i
            Required input fields are surface_air_pressure, surface_geopotential, specific_humidity ar
            on full hybrid sigma pressure levels. Note, this procedure is an approximation, which doesn'
                 account the effects of e.g. cloud ice and water, rain and snow.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`detrend` *Detrend time series*

Description

Every time series in infile is linearly detrended. For every field element x only those timesteps t belong to the sample S(x), which have i(t,x) NE miss. It is assumed that all timesteps are equidistant, if this is not the case set the parameter equal=false.

Usage

```
cdo_detrend(ifile, equal = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`equal`	BOOL - Set to false for unequal distributed timesteps (default: true)
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

Note

This operator has to keep the fields of all timesteps concurrently in the memory. If not enough memory is available use the operators trend and subtrend.

`dhourstat` *Multiday hourly statistics*

Description

This module computes statistical values of each hour of day. Depending on the chosen operator the minimum, maximum, range, sum, average, variance or standard deviation of each hour of day in infile is written to outfile. The date information in an output field is the date of the last contributing input field.

Usage

```
cdo_dhouravg(ifile, ofile = NULL)

cdo_dhourmax(ifile, ofile = NULL)

cdo_dhourmean(ifile, ofile = NULL)

cdo_dhourmin(ifile, ofile = NULL)

cdo_dhourrange(ifile, ofile = NULL)

cdo_dhourstd(ifile, ofile = NULL)

cdo_dhourstd1(ifile, ofile = NULL)

cdo_dhoursum(ifile, ofile = NULL)

cdo_dhourvar(ifile, ofile = NULL)

cdo_dhourvar1(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Details

```
dhourmin    Multi-day hourly minimum
            o(01,x) = min\{i(t,x), day(i(t)) = 01\}
                                 ...
            o(24,x) = min\{i(t,x), day(i(t)) = 24\}
dhourmax    Multi-day hourly maximum
            o(01,x) = max\{i(t,x), day(i(t)) = 01\}
                                 ...
            o(24,x) = max\{i(t,x), day(i(t)) = 24\}
```

```
dhourrange  Multi-day hourly range
            o(01,x) = range\{i(t,x), day(i(t)) = 01\}
                               ...
            o(24,x) = range\{i(t,x), day(i(t)) = 24\}
dhoursum    Multi-day hourly sum
            o(01,x) = sum\{i(t,x), day(i(t)) = 01\}
                               ...
            o(24,x) = sum\{i(t,x), day(i(t)) = 24\}
dhourmean   Multi-day hourly mean
            o(01,x) = mean\{i(t,x), day(i(t)) = 01\}
                               ...
            o(24,x) = mean\{i(t,x), day(i(t)) = 24\}
dhouravg    Multi-day hourly average
            o(01,x) = avg\{i(t,x), day(i(t)) = 01\}
                               ...
            o(24,x) = avg\{i(t,x), day(i(t)) = 24\}
dhourstd    Multi-day hourly standard deviation
            Normalize by n.

            o(01,x) = std\{i(t,x), day(i(t)) = 01\}
                               ...
            o(24,x) = std\{i(t,x), day(i(t)) = 24\}
dhourstd1   Multi-day hourly standard deviation (n-1)
            Normalize by (n-1).

            o(01,x) = std1\{i(t,x), day(i(t)) = 01\}
                               ...
            o(24,x) = std1\{i(t,x), day(i(t)) = 24\}
dhourvar    Multi-day hourly variance
            Normalize by n.

            o(01,x) = var\{i(t,x), day(i(t)) = 01\}
                               ...
            o(24,x) = var\{i(t,x), day(i(t)) = 24\}
dhourvar1   Multi-day hourly variance (n-1)
            Normalize by (n-1).

            o(01,x) = var1\{i(t,x), day(i(t)) = 01\}
                               ...
            o(24,x) = var1\{i(t,x), day(i(t)) = 24\}
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

diff *Compare two datasets field by field*

Description

Compares the contents of two datasets field by field. The input datasets need to have the same structure and its fields need to have the dimensions. Try the option names if the number of variables differ. Exit status is 0 if inputs are the same and 1 if they differ.

Usage

```
cdo_diff(
  ifile1,
  ifile2,
  maxcount = NULL,
  abslim = NULL,
  rellim = NULL,
  names = NULL
)

cdo_diffn(
  ifile1,
  ifile2,
  maxcount = NULL,
  abslim = NULL,
  rellim = NULL,
  names = NULL
)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`maxcount`	INTEGER - Stop after maxcount different fields
`abslim`	FLOAT - Limit of the maximum absolute difference (default: 0)
`rellim`	FLOAT - Limit of the maximum relative difference (default: 1)
`names`	STRING - Consideration of the variable names of only one input file (left/right) or the intersection of both (intersect).

Details

```
diff   Compare two datasets listed by parameter id
       Provides statistics on differences between two datasets.
     For each pair of fields the operator prints one line with the following information:
       - Date and Time
       - Level, Gridsize and number of Missing values
       - Number of different values
```

```
        - Occurrence of coefficient pairs with different signs (S)
        - Occurrence of zero values (Z)
        - Maxima of absolute difference of coefficient pairs
      - Maxima of relative difference of non-zero coefficient pairs with equal signs
        - Parameter identifier
diffn   Compare two datasets listed by parameter name
        The same as operator diff. Using the name instead of the
        identifier to label the parameter.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

---

distgrid *Distribute horizontal grid*

Description

This operator distributes a dataset into smaller pieces. Each output file contains a different region of the horizontal source grid. 2D Lon/Lat grids can be split into nx*ny pieces, where a target grid region contains a structured longitude/latitude box of the source grid. Data on an unstructured grid is split into nx pieces. The output files will be named <obase><xxx><suffix> where suffix is the filename extension derived from the file format. xxx will have five digits with the number of the target region.

Usage

```
cdo_distgrid(ifile, nx = NULL, ny = NULL, obase = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`nx`	INTEGER - Number of regions in x direction, or number of pieces for unstructured grids
`ny`	INTEGER - Number of regions in y direction [default: 1]
`obase`	String with the basename of the output files.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

Note

This operator needs to open all output files simultaneously. The maximum number of open files depends on the operating system!

`dminutestat` *Multiday by the minute statistics*

Description

This module computes statistical values of each minute of day. Depending on the chosen operator the minimum, maximum, range, sum, average, variance or standard deviation of each minute of day in infile is written to outfile. The date information in an output field is the date of the last contributing input field.

Usage

```
cdo_dminuteavg(ifile, ofile = NULL)

cdo_dminutemax(ifile, ofile = NULL)

cdo_dminutemean(ifile, ofile = NULL)

cdo_dminutemin(ifile, ofile = NULL)

cdo_dminuterange(ifile, ofile = NULL)

cdo_dminutestd(ifile, ofile = NULL)

cdo_dminutestd1(ifile, ofile = NULL)

cdo_dminutesum(ifile, ofile = NULL)

cdo_dminutevar(ifile, ofile = NULL)

cdo_dminutevar1(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Details

```
dminutemin    Multi-day by the minute minimum
              o(01,x) = min\{i(t,x), day(i(t)) = 01\}
                                ...
              o(1440,x) = min\{i(t,x), day(i(t)) = 1440\}
dminutemax    Multi-day by the minute maximum
              o(01,x) = max\{i(t,x), day(i(t)) = 01\}
                                 ...
              o(1440,x) = max\{i(t,x), day(i(t)) = 1440\}
dminuterange  Multi-day by the minute range
              o(01,x) = range\{i(t,x), day(i(t)) = 01\}
                                 ...
              o(1440,x) = range\{i(t,x), day(i(t)) = 1440\}
dminutesum    Multi-day by the minute sum
              o(01,x) = sum\{i(t,x), day(i(t)) = 01\}
                                 ...
              o(1440,x) = sum\{i(t,x), day(i(t)) = 1440\}
dminutemean   Multi-day by the minute mean
              o(01,x) = mean\{i(t,x), day(i(t)) = 01\}
                                 ...
              o(1440,x) = mean\{i(t,x), day(i(t)) = 1440\}
dminuteavg    Multi-day by the minute average
              o(01,x) = avg\{i(t,x), day(i(t)) = 01\}
                                 ...
              o(1440,x) = avg\{i(t,x), day(i(t)) = 1440\}
dminutestd    Multi-day by the minute standard deviation
              Normalize by n.

              o(01,x) = std\{i(t,x), day(i(t)) = 01\}
                                 ...
              o(1440,x) = std\{i(t,x), day(i(t)) = 1440\}
dminutestd1   Multi-day by the minute standard deviation (n-1)
              Normalize by (n-1).

              o(01,x) = std1\{i(t,x), day(i(t)) = 01\}
                                 ...
              o(1440,x) = std1\{i(t,x), day(i(t)) = 1440\}
dminutevar    Multi-day by the minute variance
              Normalize by n.

              o(01,x) = var\{i(t,x), day(i(t)) = 01\}
                                 ...
              o(1440,x) = var\{i(t,x), day(i(t)) = 1440\}
dminutevar1   Multi-day by the minute variance (n-1)
              Normalize by (n-1).

              o(01,x) = var1\{i(t,x), day(i(t)) = 01\}
                                 ...
              o(1440,x) = var1\{i(t,x), day(i(t)) = 1440\}
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don’t return filenames return a character vector with the string output.

`duplicate` *Duplicates a dataset*

Description

This operator duplicates the contents of infile and writes the result to outfile. The optional parameter sets the number of duplicates, the default is 2.

Usage

```
cdo_duplicate(ifile, ndup = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ndup`	INTEGER - Number of duplicates, default is 2.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don’t return filenames return a character vector with the string output.

`ecacdd` *Consecutive dry days index per time period*

Description

Let infile be a time series of the daily precipitation amount RR, then the largest number of consecutive days where RR is less than R is counted. R is an optional parameter with default R = 1 mm. A further output variable is the number of dry periods of more than N days. Parameter is a comma-separated list of "key=values" pairs.

Usage

```
cdo_eca_cdd(ifile, R = NULL, N = NULL, freq = NULL, ofile = NULL)

cdo_etccdi_cdd(ifile, R = NULL, N = NULL, freq = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`R`	FLOAT - Precipitation threshold (unit: mm; default: R = 1 mm)
`N`	INTEGER - Minimum number of days exceeded (default: N = 5)
`freq`	STRING - Output frequency (year, month)
`ofile`	String with the path to the output file.

Details

```
eca_cdd     Consecutive dry days index per time period
            The operator counts over the entire time series.
            The date information of a timestep in outfile is the date of
            the last contributing timestep in infile.
etccdi_cdd  Consecutive dry days index per time period
            The default output frequency is yearly.
            Periods within overlapping years are accounted for the first year.
            The date information of a timestep in outfile is the mid of
            the frequency interval.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`ecacfd` *Consecutive frost days index per time period*

Description

Let infile be a time series of the daily minimum temperature TN, then the largest number of consecutive days where TN < 0 °C is counted. Note that TN have to be given in units of Kelvin. A further output variable is the number of frost periods of more than N days. The date information of a timestep in outfile is the date of the last contributing timestep in infile.

Usage

```
cdo_eca_cfd(ifile, N = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`N`	INTEGER - Minimum number of days exceeded (default: N = 5)
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

---

ecacsu *Consecutive summer days index per time period*

Description

Let infile be a time series of the daily maximum temperature TX, then the largest number of consecutive days where TX > T is counted. The number T is an optional parameter with default T = 25°C. Note that TN have to be given in units of Kelvin, whereas T have to be given in degrees Celsius. A further output variable is the number of summer periods of more than N days. The date information of a timestep in outfile is the date of the last contributing timestep in infile.

Usage

```
cdo_eca_csu(ifile, T = NULL, N = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`T`	FLOAT - Temperature threshold (unit: °C; default: T = 25°C)
`N`	INTEGER - Minimum number of days exceeded (default: N = 5)
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

ecacwd — *Consecutive wet days index per time period*

Description

Let infile be a time series of the daily precipitation amount RR, then the largest number of consecutive days where RR is at least R is counted. R is an optional parameter with default R = 1 mm. A further output variable is the number of wet periods of more than N days. Parameter is a comma-separated list of "key=values" pairs.

Usage

```
cdo_eca_cwd(ifile, R = NULL, N = NULL, freq = NULL, ofile = NULL)

cdo_etccdi_cwd(ifile, R = NULL, N = NULL, freq = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`R`	FLOAT - Precipitation threshold (unit: mm; default: R = 1 mm)
`N`	INTEGER - Minimum number of days exceeded (default: N = 5)
`freq`	STRING - Output frequency (year, month)
`ofile`	String with the path to the output file.

Details

```
eca_cwd     Consecutive wet days index per time period
            The operator counts over the entire time series.
            The date information of a timestep in outfile is the date of
            the last contributing timestep in infile.
etccdi_cwd  Consecutive wet days index per time period
            The default output frequency is yearly.
            Periods within overlapping years are accounted for the first year.
            The date information of a timestep in outfile is the mid of
            the frequency interval.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`ecacwdi` *Cold wave duration index wrt mean of reference period*

Description

Let infile1 be a time series of the daily minimum temperature TN, and let infile2 be the mean TNnorm of daily minimum temperatures for any period used as reference. Then counted is the number of days where, in intervals of at least nday consecutive days, TN < TNnorm - T. The numbers nday and T are optional parameters with default nday = 6 and T = 5°C. A further output variable is the number of cold waves longer than or equal to nday days. TNnorm is calculated as the mean of minimum temperatures of a five day window centred on each calendar day of a given climate reference period. Note that both TN and TNnorm have to be given in the same units. The date information of a timestep in outfile is the date of the last contributing timestep in infile1.

Usage

```
cdo_eca_cwdi(ifile1, ifile2, nday = NULL, T = NULL, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`nday`	INTEGER - Number of consecutive days (default: nday = 6)
`T`	FLOAT - Temperature offset (unit: °C; default: T = 5°C)
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`ecacwfi` *Coldspell days index wrt 10th percentile of reference period*

Description

Let infile1 be a time series of the daily mean temperature TG, and infile2 be the 10th percentile TGn10 of daily mean temperatures for any period used as reference. Then counted is the number of days where, in intervals of at least nday consecutive days, TG < TGn10. The number nday is an optional parameter with default nday = 6. A further output variable is the number of cold-spell periods longer than or equal to nday days. TGn10 is calculated as the 10th percentile of daily mean temperatures of a five day window centred on each calendar day of a given climate reference period. Note that both TG and TGn10 have to be given in the same units.

Usage

```
cdo_eca_cwfi(ifile1, ifile2, nday = NULL, freq = NULL, ofile = NULL)

cdo_etccdi_csdi(ifile1, ifile2, nday = NULL, freq = NULL, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`nday`	INTEGER - Number of consecutive days (default: nday = 6)
`freq`	STRING - Output frequency (year, month)
`ofile`	String with the path to the output file.

Details

```
eca_cwfi     Cold-spell days index wrt 10th percentile of reference period
             The operator counts over the entire time series.
             The date information of a timestep in outfile is the date of
             the last contributing timestep in infile.
etccdi_csdi  Cold-spell duration index
             The default output frequency is yearly.
             Periods within overlapping years are accounted for the first year.
             The date information of a timestep in outfile is the mid of
             the frequency interval.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`ecaetr` *Intraperiod extreme temperature range*

Description

Let infile1 and infile2 be time series of thr maximum and minimum temperature TX and TN, respectively. Then the extreme temperature range is the difference of the maximum of TX and the minimum of TN. Note that TX and TN have to be given in the same units. The date information of a timestep in outfile is the date of the last contributing timesteps in infile1 and infile2.

Usage

```
cdo_eca_etr(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`ecafd` *Frost days index per time period*

Description

Let infile be a time series of the daily minimum temperature TN, then the number of days where TN < 0 °C is counted. Note that TN have to be given in units of Kelvin. Parameter is a comma-separated list of "key=value" pairs.

Usage

```
cdo_eca_fd(ifile, freq = NULL, ofile = NULL)

cdo_etccdi_fd(ifile, freq = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`freq`	STRING - Output frequency (year, month)
`ofile`	String with the path to the output file.

Details

```
eca_fd     Frost days index per time period
           The operator counts over the entire time series.
           The date information of a timestep in outfile is the date of
           the last contributing timestep in infile.
etccdi_fd  Frost days index per time period
           The default output frequency is yearly.
           The date information of a timestep in outfile is the mid of
           the frequency interval.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`ecagsl` *Thermal Growing season length index*

Description

Let infile1 be a time series of the daily mean temperature TG, and infile2 be a land-water mask. Within a period of 12 months, the thermal growing season length is officially defined as the number of days between: - first occurrence of at least nday consecutive days with TG > T - first occurrence of at least nday consecutive days with TG < T within the last 6 months On northern hemisphere, this period corresponds with the regular year, whereas on southern hemisphere, it starts at July 1st. Please note, that this definition may lead to weird results concerning values TG = T: In the first half of the period, these days do not contribute to the gsl, but they do within the second half. Moreover this definition could lead to discontinuous values in equatorial regions. The numbers nday and T are optional parameter with default nday = 6 and T = 5°C. The number fland is an optional parameter with default value fland = 0.5 and denotes the fraction of a grid point that have to be covered by land in order to be included in the calculation. A further output variable is the start day of year of the growing season. Note that TG have to be given in units of Kelvin, whereas T have to be given in degrees Celsius. The date information of a timestep in outfile is the date of the last contributing timestep in infile.

Usage

```
cdo_eca_gsl(ifile1, ifile2, nday = NULL, T = NULL, fland = NULL, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`nday`	INTEGER - Number of consecutive days (default: nday = 6)
`T`	FLOAT - Temperature threshold (unit: °C; default: T = 5°C)
`fland`	FLOAT - Land fraction threshold (default: fland = 0.5)
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`ecahd` *Heating degree days per time period*

Description

Let infile be a time series of the daily mean temperature TG, then the heating degree days are defined as the sum of T1 - TG, where only values TG < T2 are considered. If T1 and T2 are omitted, a temperature of 17°C is used for both parameters. If only T1 is given, T2 is set to T1. Note that TG have to be given in units of kelvin, whereas T1 and T2 have to be given in degrees Celsius. The date information of a timestep in outfile is the date of the last contributing timestep in infile.

Usage

```
cdo_eca_hd(ifile, T1 = NULL, T2 = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`T1`	FLOAT - Temperature limit (unit: °C; default: T1 = 17°C)
`T2`	FLOAT - Temperature limit (unit: °C; default: T2 = T1)
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`ecahwdi` *Heat wave duration index wrt mean of reference period*

Description

Let infile1 be a time series of the daily maximum temperature TX, and let infile2 be the mean TXnorm of daily maximum temperatures for any period used as reference. Then counted is the number of days where, in intervals of at least nday consecutive days, TX > TXnorm + T. The numbers nday and T are optional parameters with default nday = 6 and T = 5°C. A further output variable is the number of heat waves longer than or equal to nday days. TXnorm is calculated as the mean of maximum temperatures of a five day window centred on each calendar day of a given climate reference period. Note that both TX and TXnorm have to be given in the same units. The date information of a timestep in outfile is the date of the last contributing timestep in infile1.

Usage

```
cdo_eca_hwdi(ifile1, ifile2, nday = NULL, T = NULL, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`nday`	INTEGER - Number of consecutive days (default: nday = 6)
`T`	FLOAT - Temperature offset (unit: °C; default: T = 5°C)
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`ecahwfi` *Warm spell days index wrt 90th percentile of reference period*

Description

Let infile1 be a time series of the daily mean temperature TG, and infile2 be the 90th percentile TGn90 of daily mean temperatures for any period used as reference. Then counted is the number of days where, in intervals of at least nday consecutive days, TG > TGn90. The number nday is an optional parameter with default nday = 6. A further output variable is the number of warm-spell periods longer than or equal to nday days. TGn90 is calculated as the 90th percentile of daily mean temperatures of a five day window centred on each calendar day of a given climate reference period. Note that both TG and TGn90 have to be given in the same units. Parameter is a comma-separated list of "key=values" pairs.

Usage

```
cdo_eca_hwfi(ifile1, ifile2, nday = NULL, freq = NULL, ofile = NULL)

cdo_etccdi_wsdi(ifile1, ifile2, nday = NULL, freq = NULL, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`nday`	INTEGER - Number of consecutive days (default: nday = 6)
`freq`	STRING - Output frequency (year, month)
`ofile`	String with the path to the output file.

Details

```
eca_hwfi     Warm spell days index wrt 90th percentile of reference period
             The operator counts over the entire time series.
             The date information of a timestep in outfile is the date of
             the last contributing timestep in infile.
etccdi_wsdi  Warm Spell Duration Index
             The default output frequency is yearly.
             Periods within overlapping years are accounted for the first year.
             The date information of a timestep in outfile is the mid of
             the frequency interval.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

---

ecaid *Ice days index per time period*

Description

Let infile be a time series of the daily maximum temperature TX, then the number of days where TX < 0 °C is counted. Note that TX have to be given in units of Kelvin. Parameter is a comma-separated list of "key=values" pairs.

Usage

```
cdo_eca_id(ifile, freq = NULL, ofile = NULL)

cdo_etccdi_id(ifile, freq = NULL, ofile = NULL)
```

Arguments

ifile	String with the path to the input file.
freq	STRING - Output frequency (year, month)
ofile	String with the path to the output file.

Details

```
eca_id     Ice days index per time period
           The operator counts over the entire time series.
           The date information of a timestep in outfile is the date of
           the last contributing timestep in infile.
etccdi_id  Ice days index per time period
           The default output frequency is yearly.
           The date information of a timestep in outfile is the mid of
           the frequency interval.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

---

ecapd *Precipitation days index per time period*

Description

Let infile be a time series of the daily precipitation amount RR in [mm] (or alternatively in [kg m-2]), then the number of days where RR is at least x mm is counted. eca_r10mm and eca_r20mm are specific ECA operators with a daily precipitation amount of 10 and 20 mm respectively. The date information of a timestep in outfile is the date of the last contributing timestep in infile except for the etccdi operator. Parameter is a comma-separated list of "key=values" pairs.

Usage

```
cdo_eca_pd(ifile, x = NULL, freq = NULL, ofile = NULL)

cdo_eca_r10mm(ifile, x = NULL, freq = NULL, ofile = NULL)

cdo_eca_r20mm(ifile, x = NULL, freq = NULL, ofile = NULL)

cdo_etccdi_r1mm(ifile, x = NULL, freq = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`x`	FLOAT - Daily precipitation amount threshold in [mm]
`freq`	STRING - Output frequency (year, month)
`ofile`	String with the path to the output file.

Details

```
eca_pd       Precipitation days index per time period
             Generic ECA operator with daily precipitation sum exceeding x mm.
eca_r10mm    Heavy precipitation days index per time period
            Specific ECA operator with daily precipitation sum exceeding 10 mm.
eca_r20mm    Very heavy precipitation days index per time period
            Specific ECA operator with daily precipitation sum exceeding 20 mm.
etccdi_r1mm  Precipitation days index per time period
             The default output frequency is yearly.
             The date information of a timestep in outfile is the mid of
             the frequency interval.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

Note

Precipitation rates in [mm/s] have to be converted to precipitation amounts (multiply with 86400 s). Apart from metadata information the result of eca_pd,1 and eca_rr1 is the same.

`ecar75p` *Moderate wet days wrt 75th percentile of reference period*

Description

Let infile1 be a time series RR of the daily precipitation amount at wet days (precipitation >= 1 mm) and infile2 be the 75th percentile RRn75 of the daily precipitation amount at wet days for any period used as reference. Then the percentage of wet days with RR > RRn75 is calculated. RRn75 is calculated as the 75th percentile of all wet days of a given climate reference period. Usually infile2 is generated by the operator ydaypctl,75. The date information of a timestep in outfile is the date of the last contributing timestep in infile1.

Usage

```
cdo_eca_r75p(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`ecar75ptot` *Precipitation percent due to R75p days*

Description

Let infile1 be a time series RR of the daily precipitation amount at wet days (precipitation >= 1 mm) and infile2 be the 75th percentile RRn75 of the daily precipitation amount at wet days for any period used as reference. Then the ratio of the precipitation sum at wet days with RR > RRn75 to the total precipitation sum is calculated. RRn75 is calculated as the 75th percentile of all wet days of a given climate reference period. Usually infile2 is generated by the operator ydaypctl,75. The date information of a timestep in outfile is the date of the last contributing timestep in infile1.

Usage

```
cdo_eca_r75ptot(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`ecar90p` *Wet days wrt 90th percentile of reference period*

Description

Let infile1 be a time series RR of the daily precipitation amount at wet days (precipitation >= 1 mm) and infile2 be the 90th percentile RRn90 of the daily precipitation amount at wet days for any period used as reference. Then the percentage of wet days with RR > RRn90 is calculated. RRn90 is calculated as the 90th percentile of all wet days of a given climate reference period. Usually infile2 is generated by the operator ydaypctl,90. The date information of a timestep in outfile is the date of the last contributing timestep in infile1.

Usage

```
cdo_eca_r90p(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`ecar90ptot` *Precipitation percent due to R90p days*

Description

Let infile1 be a time series RR of the daily precipitation amount at wet days (precipitation >= 1 mm) and infile2 be the 90th percentile RRn90 of the daily precipitation amount at wet days for any period used as reference. Then the ratio of the precipitation sum at wet days with RR > RRn90 to the total precipitation sum is calculated. RRn90 is calculated as the 90th percentile of all wet days of a given climate reference period. Usually infile2 is generated by the operator ydaypctl,90. The date information of a timestep in outfile is the date of the last contributing timestep in infile1.

Usage

```
cdo_eca_r90ptot(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

ecar95p *Very wet days wrt 95th percentile of reference period*

Description

Let infile1 be a time series RR of the daily precipitation amount at wet days (precipitation >= 1 mm) and infile2 be the 95th percentile RRn95 of the daily precipitation amount at wet days for any period used as reference. Then the percentage of wet days with RR > RRn95 is calculated. RRn95 is calculated as the 95th percentile of all wet days of a given climate reference period. Usually infile2 is generated by the operator ydaypctl,95. The date information of a timestep in outfile is the date of the last contributing timestep in infile1.

Usage

```
cdo_eca_r95p(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

ecar95ptot *Precipitation percent due to R95p days*

Description

Let infile1 be a time series RR of the daily precipitation amount at wet days (precipitation >= 1 mm) and infile2 be the 95th percentile RRn95 of the daily precipitation amount at wet days for any period used as reference. Then the ratio of the precipitation sum at wet days with RR > RRn95 to the total precipitation sum is calculated. RRn95 is calculated as the 95th percentile of all wet days of a given climate reference period. Usually infile2 is generated by the operator ydaypctl,95. The date information of a timestep in outfile is the date of the last contributing timestep in infile1.

Usage

```
cdo_eca_r95ptot(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

---

ecar99p *Extremely wet days wrt 99th percentile of reference period*

Description

Let infile1 be a time series RR of the daily precipitation amount at wet days (precipitation >= 1 mm) and infile2 be the 99th percentile RRn99 of the daily precipitation amount at wet days for any period used as reference. Then the percentage of wet days with RR > RRn99 is calculated. RRn99 is calculated as the 99th percentile of all wet days of a given climate reference period. Usually infile2 is generated by the operator ydaypctl,99. The date information of a timestep in outfile is the date of the last contributing timestep in infile1.

Usage

```
cdo_eca_r99p(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`ecar99ptot` *Precipitation percent due to R99p days*

Description

Let infile1 be a time series RR of the daily precipitation amount at wet days (precipitation >= 1 mm) and infile2 be the 99th percentile RRn99 of the daily precipitation amount at wet days for any period used as reference. Then the ratio of the precipitation sum at wet days with RR > RRn99 to the total precipitation sum is calculated. RRn99 is calculated as the 99th percentile of all wet days of a given climate reference period. Usually infile2 is generated by the operator ydaypctl,99. The date information of a timestep in outfile is the date of the last contributing timestep in infile1.

Usage

```
cdo_eca_r99ptot(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`ecarr1` *Wet days index per time period*

Description

Let infile be a time series of the daily precipitation amount RR in [mm] (or alternatively in [kg m-2]), then the number of days where RR is at least R is counted. R is an optional parameter with default R = 1 mm. The date information of a timestep in outfile is the date of the last contributing timestep in infile.

Usage

```
cdo_eca_rr1(ifile, R = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`R`	FLOAT - Precipitation threshold (unit: mm; default: R = 1 mm)
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

---

ecarx1day *Highest one day precipitation amount per time period*

Description

Let infile be a time series of the daily precipitation amount RR, then the maximum of RR is written to outfile. If the optional parameter mode is set to 'm' the maximum daily precipitation amounts are determined for each month. Parameter is a comma-separated list of "key=values" pairs.

Usage

```
cdo_eca_rx1day(ifile, freq = NULL, ofile = NULL)

cdo_etccdi_rx1day(ifile, freq = NULL, ofile = NULL)
```

Arguments

ifile	String with the path to the input file.
freq	STRING - Output frequency (year, month)
ofile	String with the path to the output file.

Details

```
eca_rx1day     Highest one day precipitation amount per time period
               The operator counts over the entire time series.
               The date information of a timestep in outfile is the date of
               the last contributing timestep in infile.
etccdi_rx1day  Maximum 1-day Precipitation
               The default output frequency is yearly.
               The date information of a timestep in outfile is the mid of
               the frequency interval.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

ecarx5day *Highest fiveday precipitation amount per time period*

Description

Let infile be a time series of 5-day precipitation totals RR, then the maximum of RR is written to outfile. A further output variable is the number of 5 day period with precipitation totals greater than x mm, where x is an optional parameter with default x = 50 mm. Parameter is a comma-separated list of "key=values" pairs.

Usage

```
cdo_eca_rx5day(ifile, x = NULL, freq = NULL, ofile = NULL)

cdo_etccdi_rx5day(ifile, x = NULL, freq = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`x`	FLOAT - Precipitation threshold (unit: mm; default: x = 50 mm)
`freq`	STRING - Output frequency (year, month)
`ofile`	String with the path to the output file.

Details

```
eca_rx5day     Highest five-day precipitation amount per time period
               The operator counts over the entire time series.
               The date information of a timestep in outfile is the date of
               the last contributing timestep in infile.
etccdi_rx5day  Highest five-day precipitation amount per time period
               The default output frequency is yearly.
              Periods within overlapping years are accounted for the first year.
               The date information of a timestep in outfile is the mid of
               the frequency interval.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don’t return filenames return a character vector with the string output.

ecasdii *Simple daily intensity index per time period*

Description

Let infile be a time series of the daily precipitation amount RR, then the mean precipitation amount at wet days (RR >= R) is written to outfile. R is an optional parameter with default R = 1 mm. The date information of a timestep in outfile is the date of the last contributing timestep in infile.

Usage

```
cdo_eca_sdii(ifile, R = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`R`	FLOAT - Precipitation threshold (unit: mm; default: R = 1 mm)
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

ecasu *Summer days index per time period*

Description

Let infile be a time series of the daily maximum temperature TX, then the number of days where TX > T is counted. The number T is an optional parameter with default T = 25°C. Note that TX have to be given in units of Kelvin, whereas T have to be given in degrees Celsius. Parameter is a comma-separated list of "key=values" pairs.

Usage

```
cdo_eca_su(ifile, T = NULL, freq = NULL, ofile = NULL)

cdo_etccdi_su(ifile, T = NULL, freq = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`T`	FLOAT - Temperature threshold (unit: °C; default: T = 25°C)
`freq`	STRING - Output frequency (year, month)
`ofile`	String with the path to the output file.

Details

```
eca_su     Summer days index per time period
           The operator counts over the entire time series.
           The date information of a timestep in outfile is the date of
           the last contributing timestep in infile.
etccdi_su  Summer days index per time period
           The default output frequency is yearly.
           The date information of a timestep in outfile is the mid of
           the frequency interval.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`ecatg10p` *Cold days percent wrt 10th percentile of reference period*

Description

Let infile1 be a time series of the daily mean temperature TG, and infile2 be the 10th percentile TGn10 of daily mean temperatures for any period used as reference. Then the percentage of time where TG < TGn10 is calculated. TGn10 is calculated as the 10th percentile of daily mean temperatures of a five day window centred on each calendar day of a given climate reference period. Note that both TG and TGn10 have to be given in the same units. The date information of a timestep in outfile is the date of the last contributing timestep in infile1.

Usage

```
cdo_eca_tg10p(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`ecatg90p` *Warm days percent wrt 90th percentile of reference period*

Description

Let infile1 be a time series of the daily mean temperature TG, and infile2 be the 90th percentile TGn90 of daily mean temperatures for any period used as reference. Then the percentage of time where TG > TGn90 is calculated. TGn90 is calculated as the 90th percentile of daily mean temperatures of a five day window centred on each calendar day of a given climate reference period. Note that both TG and TGn90 have to be given in the same units. The date information of a timestep in outfile is the date of the last contributing timestep in infile1.

Usage

```
cdo_eca_tg90p(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`ecatn10p` *Cold nights percent wrt 10th percentile of reference period*

Description

Let infile1 be a time serie of the daily minimum temperature TN, and infile2 be the 10th percentile TNn10 of daily minimum temperatures for any period used as reference. Then the percentage of time where TN < TNn10 is calculated. TNn10 is calculated as the 10th percentile of daily minimum temperatures of a five day window centred on each calendar day of a given climate reference period. Note that both TN and TNn10 have to be given in the same units. The date information of a timestep in outfile is the date of the last contributing timestep in infile1.

Usage

```
cdo_eca_tn10p(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`ecatn90p` *Warm nights percent wrt 90th percentile of reference period*

Description

Let infile1 be a time series of the daily minimum temperature TN, and infile2 be the 90th percentile TNn90 of daily minimum temperatures for any period used as reference. Then the percentage of time where TN > TNn90 is calculated. TNn90 is calculated as the 90th percentile of daily minimum temperatures of a five day window centred on each calendar day of a given climate reference period. Note that both TN and TNn90 have to be given in the same units. The date information of a timestep in outfile is the date of the last contributing timestep in infile1.

Usage

```
cdo_eca_tn90p(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

ecatr *Tropical nights index per time period*

Description

Let infile be a time series of the daily minimum temperature TN, then the number of days where TN > T is counted. The number T is an optional parameter with default T = 20°C. Note that TN have to be given in units of Kelvin, whereas T have to be given in degrees Celsius. Parameter is a comma-separated list of "key=values" pairs.

Usage

```
cdo_eca_tr(ifile, T = NULL, freq = NULL, ofile = NULL)

cdo_etccdi_tr(ifile, T = NULL, freq = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`T`	FLOAT - Temperature threshold (unit: °C; default: T = 20°C)
`freq`	STRING - Output frequency (year, month)
`ofile`	String with the path to the output file.

Details

```
eca_tr     Tropical nights index per time period
           The operator counts over the entire time series.
           The date information of a timestep in outfile is the date of
           the last contributing timestep in infile.
etccdi_tr  Tropical nights index per time period
           The default output frequency is yearly.
           The date information of a timestep in outfile is the mid of
           the frequency interval.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don’t return filenames return a character vector with the string output.

ecatx10p *Very cold days percent wrt 10th percentile of reference period*

Description

Let infile1 be a time series of the daily maximum temperature TX, and infile2 be the 10th percentile TXn10 of daily maximum temperatures for any period used as reference. Then the percentage of time where TX < TXn10. is calculated. TXn10 is calculated as the 10th percentile of daily maximum temperatures of a five day window centred on each calendar day of a given climate reference period. Note that both TX and TXn10 have to be givenin the same units. The date information of a timestep in outfile is the date of the last contributing timestep in infile1.

Usage

```
cdo_eca_tx10p(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

ecatx90p *Very warm days percent wrt 90th percentile of reference period*

Description

Let infile1 be a time series of the daily maximum temperature TX, and infile2 be the 90th percentile TXn90 of daily maximum temperatures for any period used as reference. Then the percentage of time where TX > TXn90. is calculated. TXn90 is calculated as the 90th percentile of daily maximum temperatures of a five day window centred on each calendar day of a given climate reference period. Note that both TX and TXn90 have to be given in the same units. The date information of a timestep in outfile is the date of the last contributing timestep in infile1.

Usage

```
cdo_eca_tx90p(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`enlarge` *Enlarge fields*

Description

Enlarge all fields of infile to a user given horizontal grid. Normally only the last field element is used for the enlargement. If however the input and output grid are regular lon/lat grids, a zonal or meridional enlargement is possible. Zonal enlargement takes place, if the xsize of the input field is 1 and the ysize of both grids are the same. For meridional enlargement the ysize have to be 1 and the xsize of both grids should have the same size.

Usage

```
cdo_enlarge(ifile, grid = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`grid`	STRING - Target grid description file or name
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`ensstat` *Statistical values over an ensemble*

Description

This module computes statistical values over an ensemble of input files. Depending on the chosen operator, the minimum, maximum, range, sum, average, standard deviation, variance, skewness, kurtosis, median or a certain percentile over all input files is written to outfile. All input files need to have the same structure with the same variables. The date information of a timestep in outfile is the date of the first input file.

Usage

```
cdo_ensavg(ifiles, p = NULL, ofile = NULL)

cdo_enskurt(ifiles, p = NULL, ofile = NULL)

cdo_ensmax(ifiles, p = NULL, ofile = NULL)

cdo_ensmean(ifiles, p = NULL, ofile = NULL)

cdo_ensmedian(ifiles, p = NULL, ofile = NULL)

cdo_ensmin(ifiles, p = NULL, ofile = NULL)

cdo_enspctl(ifiles, p = NULL, ofile = NULL)

cdo_ensrange(ifiles, p = NULL, ofile = NULL)

cdo_ensskew(ifiles, p = NULL, ofile = NULL)

cdo_ensstd(ifiles, p = NULL, ofile = NULL)

cdo_ensstd1(ifiles, p = NULL, ofile = NULL)

cdo_enssum(ifiles, p = NULL, ofile = NULL)

cdo_ensvar(ifiles, p = NULL, ofile = NULL)

cdo_ensvar1(ifiles, p = NULL, ofile = NULL)
```

Arguments

`ifiles`	Character vector with the path to the input files.
`p`	FLOAT - Percentile number in {0, ..., 100}
`ofile`	String with the path to the output file.

Details

```
ensmin     Ensemble minimum
           o(t,x) = min\{i1(t,x), i2(t,x), ..., in(t,x)\}
ensmax     Ensemble maximum
           o(t,x) = max\{i1(t,x), i2(t,x), ..., in(t,x)\}
ensrange   Ensemble range
           o(t,x) = range\{i1(t,x), i2(t,x), ..., in(t,x)\}
enssum     Ensemble sum
           o(t,x) = sum\{i1(t,x), i2(t,x), ..., in(t,x)\}
ensmean    Ensemble mean
           o(t,x) = mean\{i1(t,x), i2(t,x), ..., in(t,x)\}
ensavg     Ensemble average
           o(t,x) = avg\{i1(t,x), i2(t,x), ..., in(t,x)\}
ensstd     Ensemble standard deviation
           Normalize by n.

           o(t,x) = std\{i1(t,x), i2(t,x), ..., in(t,x)\}
ensstd1    Ensemble standard deviation (n-1)
           Normalize by (n-1).

           o(t,x) = std1\{i1(t,x), i2(t,x), ..., in(t,x)\}
ensvar     Ensemble variance
           Normalize by n.

           o(t,x) = var\{i1(t,x), i2(t,x), ..., in(t,x)\}
ensvar1    Ensemble variance (n-1)
           Normalize by (n-1).

           o(t,x) = var1\{i1(t,x), i2(t,x), ..., in(t,x)\}
ensskew    Ensemble skewness
           o(t,x) = skew\{i1(t,x), i2(t,x), ..., in(t,x)\}
enskurt    Ensemble kurtosis
           o(t,x) = kurt\{i1(t,x), i2(t,x), ..., in(t,x)\}
ensmedian  Ensemble median
           o(t,x) = median\{i1(t,x), i2(t,x), ..., in(t,x)\}
enspctl    Ensemble percentiles
           o(t,x) = pth percentile \{i1(t,x), i2(t,x), ..., in(t,x)\}
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

Note

Operators of this module need to open all input files simultaneously. The maximum number of open files depends on the operating system!

`ensstat2` *Statistical values over an ensemble*

Description

This module computes statistical values over the ensemble of ensfiles using obsfile as a reference. Depending on the operator a ranked Histogram or a roc-curve over all Ensembles ensfiles with reference to obsfile is written to outfile. The date and grid information of a timestep in outfile is the date of the first input file. Thus all input files are required to have the same structure in terms of the gridsize, variable definitions and number of timesteps. All Operators in this module use obsfile as the reference (for instance an observation) whereas ensfiles are understood as an ensemble consisting of n (where n is the number of ensfiles) members. The operators ensrkhistspace and ensrkhisttime compute Ranked Histograms. Therefor the vertical axis is utilized as the Histogram axis, which prohibits the use of files containing more than one level. The histogram axis has nensfiles+1 bins with level 0 containing for each grid point the number of observations being smaller as all ensembles and level nensfiles+1 indicating the number of observations being larger than all ensembles. ensrkhisttime computes a ranked histogram at each timestep reducing each horizontal grid to a 1x1 grid and keeping the time axis as in obsfile. Contrary ensrkhistspace computes a histogram at each grid point keeping the horizontal grid for each variable and reducing the time-axis. The time information is that from the last timestep in obsfile.

Usage

```
cdo_ensrkhistspace(ifiles, ofile = NULL)

cdo_ensrkhisttime(ifiles, ofile = NULL)

cdo_ensroc(ifiles, ofile = NULL)
```

Arguments

`ifiles`	Character vector with the path to the input files.
`ofile`	String with the path to the output file.

Details

```
ensrkhistspace  Ranked Histogram averaged over space
ensrkhisttime   Ranked Histogram averaged over time
ensroc          Ensemble Receiver Operating characteristics
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`ensval`	*Ensemble validation tools*

Description

This module computes ensemble validation scores and their decomposition such as the Brier and cumulative ranked probability score (CRPS). The first file is used as a reference it can be a climatology, observation or reanalysis against which the skill of the ensembles given in infiles is measured. Depending on the operator a number of output files is generated each containing the skill score and its decomposition corresponding to the operator. The output is averaged over horizontal fields using appropriate weights for each level and timestep in rfile. All input files need to have the same structure with the same variables. The date information of a timestep in outfile is the date of the first input file. The output files are named as <outfilebase>.<type>.<filesuffix> where <type> depends on the operator and <filesuffix> is determined from the output file type. There are three output files for operator enscrps and four output files for operator ensbrs. The CRPS and its decomposition into Reliability and the potential CRPS are calculated by an appropriate averaging over the field members (note, that the CRPS does *not* average linearly). In the three output files <type> has the following meaning: crps for the CRPS, reli for the reliability and crpspot for the potential crps. The relation CRPS = CRPS_{pot} + RELI holds. The Brier score of the Ensemble given by infiles with respect to the reference given in rfile and the threshold x is calculated. In the four output files <type> has the following meaning: brs for the Brier score wrt threshold x; brsreli for the Brier score reliability wrt threshold x; brsreso for the Brier score resolution wrt threshold x; brsunct for the Brier score uncertainty wrt threshold x. In analogy to the CRPS the following relation holds: BRS(x) = RELI(x)-RESO(x)+ UNCT(x). The implementation of the decomposition of the CRPS and Brier Score follows Hans Hersbach (2000): Decomposition of the Continuous Ranked Probability Score for Ensemble Prediction Systems, in: Weather and Forecasting (15) pp. 559-570. The CRPS code decomposition has been verified against the CRAN - ensemble validation package from R. Differences occur when grid-cell area is not uniform as the implementation in R does not account for that.

Usage

```
cdo_ensbrs(ifiles, obase = NULL)

cdo_enscrps(ifiles, obase = NULL)
```

Arguments

`ifiles`	Character vector with the path to the input files.
`obase`	String with the basename of the output files.

Details

```
enscrps  Ensemble CRPS and decomposition
ensbrs   Ensemble Brier score
         Ensemble Brier Score and Decomposition
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`eofcoeff` *Principal coefficients of EOFs*

Description

This module calculates the time series of the principal coefficients for given EOF (empirical orthogonal functions) and data. Time steps in infile1 are assumed to be the EOFs, time steps in infile2 are assumed to be the time series. Note, that this operator calculates a non weighted dot product of the fields in infile1 and infile2. For consistency set the environment variable CDO_WEIGHT_MODE=off when using eof or eof3d. There will be a separate file containing a time series of principal coefficients with time information from infile2 for each EOF in infile1. Output files will be numbered as <obase><neof><suffix> where neof+1 is the number of the EOF (timestep) in infile1 and suffix is the filename extension derived from the file format.

Usage

```
cdo_eofcoeff(ifile1, ifile2, obase = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`obase`	String with the basename of the output files.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

eofs *Empirical Orthogonal Functions*

Description

This module calculates empirical orthogonal functions of the data in infile as the eigen values of the scatter matrix (covariance matrix) S of the data sample z(t). A more detailed description can be found above. Please note, that the input data are assumed to be anomalies. If operator eof is chosen, the EOFs are computed in either time or spatial space, whichever is the fastest. If the user already knows, which computation is faster, the module can be forced to perform a computation in time- or gridspace by using the operators eoftime or eofspatial, respectively. This can enhance performance, especially for very long time series, where the number of timesteps is larger than the number of grid-points. Data in infile are assumed to be anomalies. If they are not, the behavior of this module is not well defined. After execution outfile1 will contain all eigen-values and outfile2 the eigenvectors e_j. All EOFs and eigen-values are computed. However, only the first neof EOFs are written to outfile2. Nonetheless, outfile1 contains all eigen-values. Missing values are not fully supported. Support is only checked for non-changing masks of missing values in time. Although there still will be results, they are not trustworthy, and a warning will occur. In the latter case we suggest to replace missing values by 0 in infile.

Usage

```
cdo_eof(ifile, neof = NULL, ofile1 = NULL, ofile2 = NULL)

cdo_eof3d(ifile, neof = NULL, ofile1 = NULL, ofile2 = NULL)

cdo_eofspatial(ifile, neof = NULL, ofile1 = NULL, ofile2 = NULL)

cdo_eoftime(ifile, neof = NULL, ofile1 = NULL, ofile2 = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`neof`	INTEGER - Number of eigen functions
`ofile1`, `ofile2`	Strings with the path to the output files.

Details

```
eof         Calculate EOFs in spatial or time space
eoftime     Calculate EOFs in time space
eofspatial  Calculate EOFs in spatial space
eof3d       Calculate 3-Dimensional EOFs in time space
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`expr` *Evaluate expressions*

Description

This module arithmetically processes every timestep of the input dataset. Each individual assignment statement have to end with a semi-colon. The special key *ALL* is used as a template. A statement with a template is replaced for all variable names. Unlike regular variables, temporary variables are never written to the output stream. To define a temporary variable simply prefix the variable name with an underscore (e.g. _varname) when the variable is declared. The following operators are supported: Operator & Meaning & Example & Result = & assignment & x = y & Assigns y to x + & addition & x + y & Sum of x and y - & subtraction & x - y & Difference of x and y * & multiplication & x * y & Product of x and y / & division & x / y & Quotient of x and y ^ & exponentiation & x ^y & Exponentiates x with y == & equal to & x == y & 1, if x equal to y; else 0 != & not equal to & x != y & 1, if x not equal to y; else 0 > & greater than & x > y & 1, if x greater than y; else 0 < & less than & x < y & 1, if x less than y; else 0 >= & greater equal & x >= y & 1, if x greater equal y; else 0 <= & less equal & x <= y & 1, if x less equal y; else 0 <=> & less equal greater & x <=> y & -1, if x less y; 1, if x greater y; else 0 && & logical AND & x && y & 1, if x and y not equal 0; else 0 || & logical OR & x || y & 1, if x or y not equal 0; else 0 ! & logical NOT & !x & 1, if x equal 0; else 0 ?: & ternary conditional & x ? y : z & y, if x not equal 0, else z The following functions are supported: Math intrinsics: abs(x) " " Absolute value of x floor(x) " " Round to largest integral value not greater than x ceil(x) " " Round to smallest integral value not less than x float(x) " " 32-bit float value of x int(x) " " Integer value of x nint(x) " " Nearest integer value of x sqr(x) " " Square of x sqrt(x) " " Square Root of x exp(x) " " Exponential of x ln(x) " " Natural logarithm of x log10(x) " " Base 10 logarithm of x sin(x) " " Sine of x, where x is specified in radians cos(x) " " Cosine of x, where x is specified in radians tan(x) " " Tangent of x, where x is specified in radians asin(x) " " Arc-sine of x, where x is specified in radians acos(x) " " Arc-cosine of x, where x is specified in radians atan(x) " " Arc-tangent of x, where x is specified in radians sinh(x) " " Hyperbolic sine of x, where x is specified in radians cosh(x) " " Hyperbolic cosine of x, where x is specified in radians tanh(x) " " Hyperbolic tangent of x, where x is specified in radians asinh(x) " " Inverse hyperbolic sine of x, where x is specified in radians acosh(x) " " Inverse hyperbolic cosine of x, where x is specified in radians atanh(x) " " Inverse hyperbolic tangent of x, where x is specified in radians rad(x) " " Convert x from degrees to radians deg(x) " " Convert x from radians to degrees rand(x) " " Replace x by pseudo-random numbers in the range of 0 to 1 isMissval(x)" " Returns 1 where x is missing mod(x,y) " " Floating-point remainder of x/ y min(x,y) " " Minimum value of x and y max(x,y) " " Maximum value of x and y pow(x,y) " " Power function hypot(x,y) " " Euclidean distance function, sqrt(x*x* + *y*y) atan2(x,y) " " Arc tangent function of y/x, using signs to determine quadrants Coordinates: clon(x) " " Longitude coordinate of x (available only if x has geographical coordinates) clat(x) " " Latitude coordinate of x (available only if x has geographical coordinates) gridarea(x) " " Grid cell area of x (available only if x has geographical coordinates) gridindex(x) " " Grid cell indices of x clev(x) " " Level coordinate of x (0, if x is a 2D surface variable) clevidx(x) " " Level index of x (0, if x is a 2D surface variable) cthickness(x)" " Layer thickness, upper minus lower level bound of x (1, if level bounds are missing) ctimestep() " " Timestep number (1 to N) cdate() " " Verification date as YYYYMMDD ctime() " " Verification time as HHMMSS.millisecond cdeltat() " " Difference between current and last timestep in seconds cday() " " Day as DD cmonth() " " Month as MM cyear() " " Year as YYYY csecond() " " Second as SS.millisecond cminute() " " Minute as MM chour() " " Hour as HH Constants: ngp(x) " " Number of horizontal grid points nlev(x) " " Number of vertical levels size(x) " " Total number of elements (ngp(x)*nlev(x)) missval(x)" " Returns the missing value of variable x Statistics over a field: fldmin(x), fldmax(x), fldrange(x), fldsum(x), fldmean(x), fldavg(x), fldstd(x), fldstd1(x), fldvar(x), fldvar1(x), fldskew(x), fldkurt(x), fldmedian(x) Zonal statistics for regular 2D grids: zonmin(x), zonmax(x), zonrange(x), zonsum(x), zonmean(x), zonavg(x), zonstd(x), zonstd1(x), zonvar(x), zonvar1(x), zonskew(x), zonkurt(x), zonmedian(x) Vertical statistics: vertmin(x), vertmax(x), vertrange(x), vertsum(x), vertmean(x), vertavg(x), vertstd(x), vertstd1(x), vertvar(x), vertvar1(x) Miscellaneous: sellevel(x,k) " " Select level k of variable x sellevidx(x,k) " " Select level index k of variable x sellevelrange(x,k1,k2) " " Select all levels of variable x in the range k1 to k2 sellevidxrange(x,k1,k2)" " Select all level indices of variable x in the range k1 to k2 remove(x) " " Remove variable x from output stream

Usage

```
cdo_aexpr(ifile, instr = NULL, filename = NULL, ofile = NULL)

cdo_aexprf(ifile, instr = NULL, filename = NULL, ofile = NULL)

cdo_expr(ifile, instr = NULL, filename = NULL, ofile = NULL)

cdo_exprf(ifile, instr = NULL, filename = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`instr`	STRING - Processing instructions (need to be 'quoted' in most cases)
`filename`	STRING - File with processing instructions
`ofile`	String with the path to the output file.

Details

```
expr    Evaluate expressions
        The processing instructions are read from the parameter.
exprf   Evaluate expressions script
        Contrary to expr the processing instructions are read from a file.
aexpr   Evaluate expressions and append results
        Same as expr, but keep input variables and append results
aexprf  Evaluate expression script and append results
        Same as exprf, but keep input variables and append results
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

Note

If the input stream contains duplicate entries of the same variable name then the last one is used.

`fdns` *Frost days where no snow index per time period*

Description

Let infile1 be a time series of the daily minimum temperature TN and infile2 be a corresponding series of daily surface snow amounts. Then the number of days where TN < 0 °C and the surface snow amount is less than 1 cm is counted. The temperature TN have to be given in units of Kelvin. The date information of a timestep in outfile is the date of the last contributing timestep in infile.

Usage

```
cdo_fdns(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`filedes` *Dataset description*

Description

This module provides operators to print meta information about a dataset. The printed meta-data depends on the chosen operator.

Usage

```
cdo_codetab(ifile)

cdo_griddes(ifile)

cdo_partab(ifile)

cdo_vct(ifile)

cdo_zaxisdes(ifile)
```

Arguments

`ifile`	String with the path to the input file.

Details

```
partab    Parameter table
          Prints all available meta information of the variables.
codetab   Parameter code table
          Prints a code table with a description of all variables.
          For each variable the operator prints one line listing the
          code, name, description and units.
griddes   Grid description
          Prints the description of all grids.
zaxisdes  Z-axis description
          Prints the description of all z-axes.
vct       Vertical coordinate table
          Prints the vertical coordinate table.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`filter` *Time series filtering*

Description

This module takes the time series for each gridpoint in infile and (fast fourier) transforms it into the frequency domain. According to the particular operator and its parameters certain frequencies are filtered (set to zero) in the frequency domain and the spectrum is (inverse fast fourier) transformed back into the time domain. To determine the frequency the time-axis of infile is used. (Data should have a constant time increment since this assumption applies for transformation. However, the time increment has to be different from zero.) All frequencies given as parameter are interpreted per year. This is done by the assumption of a 365-day calendar. Consequently if you want to perform multiyear-filtering accurately you have to delete the 29th of February. If your infile has a 360 year calendar the frequency parameters fmin respectively fmax should be multiplied with a factor of 360/365 in order to obtain accurate results. For the set up of a frequency filter the frequency parameters have to be adjusted to a frequency in the data. Here fmin is rounded down and fmax is always rounded up. Consequently it is possible to use bandpass with fmin=fmax without getting a zero-field for outfile. Hints for efficient usage: - to get reliable results the time-series has to be detrended (cdo detrend) - the lowest frequency greater zero that can be contained in infile is 1/(N*dT), - the greatest frequency is 1/(2dT) (Nyquist frequency), with N the number of timesteps and dT the time increment of infile in years. Missing value support for operators in this module is not implemented, yet!

Usage

```
cdo_bandpass(ifile, fmin = NULL, fmax = NULL, ofile = NULL)

cdo_highpass(ifile, fmin = NULL, fmax = NULL, ofile = NULL)

cdo_lowpass(ifile, fmin = NULL, fmax = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`fmin`	FLOAT Minimum - frequency per year that passes the filter.
`fmax`	FLOAT Maximum - frequency per year that passes the filter.
`ofile`	String with the path to the output file.

Details

```
bandpass  Bandpass filtering
          Bandpass filtering (pass for frequencies between fmin and fmax).
      Suppresses all variability outside the frequency range specified by \[fmin,fmax\].
lowpass   Lowpass filtering
          Lowpass filtering (pass for frequencies lower than fmax).
          Suppresses all variability with frequencies greater than fmax.
highpass  Highpass filtering
          Highpass filtering (pass for frequencies greater than fmin).
          Suppresses all variabilty with frequencies lower than fmin.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don’t return filenames return a character vector with the string output.

Note

For better performace of these operators use the CDO configure option –with-fftw3.

`fldcor` *Correlation in grid space*

Description

The correlation coefficient is a quantity that gives the quality of a least squares fitting to the original data. This operator correlates all gridpoints of two fields for each timestep. With S(t) = {x, i_1(t,x) != missval and i_2(t,x) != missval} it is o(t,1) = Cor{(i_1(t,x), i_2(t,x)), x_1 < x <= x_n} where w(x) are the area weights obtained by the input streams. For every timestep t only those field elements x belong to the sample, which have i_1(t,x) != missval and i_2(t,x) != missval.

Usage

```
cdo_fldcor(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`fldcovar` *Covariance in grid space*

Description

This operator calculates the covariance of two fields over all gridpoints for each timestep. With S(t) = {x, i_1(t,x) != missval and i_2(t,x) != missval} it is o(t,1) = Covar{(i_1(t,x), i_2(t,x)), x_1 < x <= x_n} where w(x) are the area weights obtained by the input streams. For every timestep t only those field elements x belong to the sample, which have i_1(t,x) != missval and i_2(t,x) != missval.

Usage

```
cdo_fldcovar(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`fldstat` *Statistical values over a field*

Description

This module computes statistical values of all input fields. A field is a horizontal layer of a data variable. Depending on the chosen operator, the minimum, maximum, range, sum, integral, average, standard deviation, variance, skewness, kurtosis, median or a certain percentile of the field is written to outfile.

Usage

```
cdo_fldavg(ifile, weights = NULL, p = NULL, ofile = NULL)

cdo_fldcount(ifile, weights = NULL, p = NULL, ofile = NULL)

cdo_fldint(ifile, weights = NULL, p = NULL, ofile = NULL)

cdo_fldkurt(ifile, weights = NULL, p = NULL, ofile = NULL)

cdo_fldmax(ifile, weights = NULL, p = NULL, ofile = NULL)

cdo_fldmean(ifile, weights = NULL, p = NULL, ofile = NULL)

cdo_fldmedian(ifile, weights = NULL, p = NULL, ofile = NULL)

cdo_fldmin(ifile, weights = NULL, p = NULL, ofile = NULL)

cdo_fldpctl(ifile, weights = NULL, p = NULL, ofile = NULL)

cdo_fldrange(ifile, weights = NULL, p = NULL, ofile = NULL)

cdo_fldskew(ifile, weights = NULL, p = NULL, ofile = NULL)

cdo_fldstd(ifile, weights = NULL, p = NULL, ofile = NULL)
```

```
cdo_fldstd1(ifile, weights = NULL, p = NULL, ofile = NULL)

cdo_fldsum(ifile, weights = NULL, p = NULL, ofile = NULL)

cdo_fldvar(ifile, weights = NULL, p = NULL, ofile = NULL)

cdo_fldvar1(ifile, weights = NULL, p = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`weights`	BOOL - weights=FALSE disables weighting by grid cell area [default: weights=TRUE]
`p`	FLOAT - Percentile number in {0, ..., 100}
`ofile`	String with the path to the output file.

Details

```
fldmin     Field minimum
           For every gridpoint x_1, ..., x_n of the same field it is:

           o(t,1) = min\{i(t,x'), x_1&lt;x'&lt;=x_n\}
fldmax     Field maximum
           For every gridpoint x_1, ..., x_n of the same field it is:

           o(t,1) = max\{i(t,x'), x_1&lt;x'&lt;=x_n\}
fldrange   Field range
           For every gridpoint x_1, ..., x_n of the same field it is:

           o(t,1) = range\{i(t,x'), x_1&lt;x'&lt;=x_n\}
fldsum     Field sum
           For every gridpoint x_1, ..., x_n of the same field it is:

           o(t,1) = sum\{i(t,x'), x_1&lt;x'&lt;=x_n\}
fldint     Field integral
           For every gridpoint x_1, ..., x_n of the same field it is:

           o(t,1) = sum\{i(t,x')*cellarea(x'), x_1&lt;x'&lt;=x_n\}
fldmean    Field mean
           For every gridpoint x_1, ..., x_n of the same field it is:

           o(t,1) = mean\{i(t,x'), x_1&lt;x'&lt;=x_n\}
           weighted by area weights obtained by the input field.
fldavg     Field average
           For every gridpoint x_1, ..., x_n of the same field it is:

           o(t,1) = avg\{i(t,x'), x_1&lt;x'&lt;=x_n\}
           weighted by area weights obtained by the input field.
fldstd     Field standard deviation
```

```
          Normalize by n. For every gridpoint x_1, ..., x_n of the same field it is:

            o(t,1) = std\{i(t,x'), x_1&lt;x'&lt;=x_n\}
            weighted by area weights obtained by the input field.
fldstd1     Field standard deviation (n-1)
          Normalize by (n-1). For every gridpoint x_1, ..., x_n of the same field it is:

            o(t,1) = std1\{i(t,x'), x_1&lt;x'&lt;=x_n\}
            weighted by area weights obtained by the input field.
fldvar      Field variance
          Normalize by n. For every gridpoint x_1, ..., x_n of the same field it is:

            o(t,1) = var\{i(t,x'), x_1&lt;x'&lt;=x_n\}
            weighted by area weights obtained by the input field.
fldvar1     Field variance (n-1)
          Normalize by (n-1). For every gridpoint x_1, ..., x_n of the same field it is:

            o(t,1) = var1\{i(t,x'), x_1&lt;x'&lt;=x_n\}
            weighted by area weights obtained by the input field.
fldskew     Field skewness
            For every gridpoint x_1, ..., x_n of the same field it is:

            o(t,1) = skew\{i(t,x'), x_1&lt;x'&lt;=x_n\}
fldkurt     Field kurtosis
            For every gridpoint x_1, ..., x_n of the same field it is:

            o(t,1) = kurt\{i(t,x'), x_1&lt;x'&lt;=x_n\}
fldmedian   Field median
            For every gridpoint x_1, ..., x_n of the same field it is:

            o(t,1) = median\{i(t,x'), x_1&lt;x'&lt;=x_n\}
fldcount    Field count
            Number of non-missing values of the field.
fldpctl     Field percentiles
            For every gridpoint x_1, ..., x_n of the same field it is:

            o(t,1) = pth percentile \{i(t,x'), x_1&lt;x'&lt;=x_n\}
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`fourier` *Fourier transformation*

Description

The fourier operator performs the fourier transformation or the inverse fourier transformation of all input fields. If the number of timesteps is a power of 2 then the algorithm of the Fast Fourier Transformation (FFT) is used. If the input stream infile consists only of complex fields, then the fields of outfile, computed by cdo -f ext fourier,1 -fourier,-1 infile outfile are the same than that of infile. For real input files see function retocomplex.

Usage

```
cdo_fourier(ifile, epsilon = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`epsilon`	INTEGER - -1: forward transformation; 1: backward transformation
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

Note

Complex numbers can only be stored in NetCDF4 and EXTRA format.

getgridcell *Get grid cell index*

Description

Get the grid cell index of one grid point selected by the parameter lon and lat.

Usage

```
cdo_gridcellindex(ifile, lon = NULL, lat = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`lon`	INTEGER - Longitude of the grid cell in degree
`lat`	INTEGER - Latitude of the grid cell in degree

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`gradsdes` *GrADS data descriptor file*

Description

Creates a GrADS data descriptor file. Supported file formats are GRIB1, NetCDF, SERVICE, EXTRA and IEG. For GRIB1 files the GrADS map file is also generated. For SERVICE and EXTRA files the grid have to be specified with the CDO option '-g <grid>'. This module takes infile in order to create filenames for the descriptor (infile.ctl) and the map (infile.gmp) file.

Usage

```
cdo_gradsdes(ifile, mapversion = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`mapversion`	INTEGER - Format version of the GrADS map file for GRIB1 datasets. Use 1 for a machinespecific version 1 GrADS map file, 2 for a machine independent version 2 GrADS map fileand 4 to support GRIB files >2GB.A version 2 map file can be used only with GrADS version 1.8 or newer.A version 4 map file can be used only with GrADS version 2.0 or newer.The default is 4 for files >2GB, otherwise 2.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`gridboxstat` *Statistical values over grid boxes*

Description

This module computes statistical values over surrounding grid boxes. Depending on the chosen operator, the minimum, maximum, range, sum, average, standard deviation, variance, skewness, kurtosis or median of the neighboring grid boxes is written to outfile. All gridbox operators only work on quadrilateral curvilinear grids.

Usage

```
cdo_gridboxavg(ifile, nx = NULL, ny = NULL, ofile = NULL)

cdo_gridboxkurt(ifile, nx = NULL, ny = NULL, ofile = NULL)

cdo_gridboxmax(ifile, nx = NULL, ny = NULL, ofile = NULL)

cdo_gridboxmean(ifile, nx = NULL, ny = NULL, ofile = NULL)

cdo_gridboxmedian(ifile, nx = NULL, ny = NULL, ofile = NULL)

cdo_gridboxmin(ifile, nx = NULL, ny = NULL, ofile = NULL)

cdo_gridboxrange(ifile, nx = NULL, ny = NULL, ofile = NULL)

cdo_gridboxskew(ifile, nx = NULL, ny = NULL, ofile = NULL)

cdo_gridboxstd(ifile, nx = NULL, ny = NULL, ofile = NULL)

cdo_gridboxstd1(ifile, nx = NULL, ny = NULL, ofile = NULL)

cdo_gridboxsum(ifile, nx = NULL, ny = NULL, ofile = NULL)

cdo_gridboxvar(ifile, nx = NULL, ny = NULL, ofile = NULL)

cdo_gridboxvar1(ifile, nx = NULL, ny = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`nx`	INTEGER - Number of grid boxes in x direction
`ny`	INTEGER - Number of grid boxes in y direction
`ofile`	String with the path to the output file.

Details

`gridboxmin`	`Gridbox minimum`
	`Minimum value of the selected grid boxes.`

`gridboxmax`	`Gridbox maximum`
	`Maximum value of the selected grid boxes.`

`gridboxrange`	`Gridbox range`
	`Range (max-min value) of the selected grid boxes.`

`gridboxsum`	`Gridbox sum`
	`Sum of the selected grid boxes.`

`gridboxmean`	`Gridbox mean`
	`Mean of the selected grid boxes.`

`gridboxavg`	`Gridbox average`
	`Average of the selected grid boxes.`

`gridboxstd`	`Gridbox standard deviation`
	`Standard deviation of the selected grid boxes. Normalize by n.`

`gridboxstd1`	`Gridbox standard deviation (n-1)`
	`Standard deviation of the selected grid boxes. Normalize by (n-1).`

`gridboxvar`	`Gridbox variance`
	`Variance of the selected grid boxes. Normalize by n.`

`gridboxvar1`	`Gridbox variance (n-1)`
	`Variance of the selected grid boxes. Normalize by (n-1).`

`gridboxskew`	`Gridbox skewness`
	`Skewness of the selected grid boxes.`

`gridboxkurt`	`Gridbox kurtosis`
	`Kurtosis of the selected grid boxes.`

`gridboxmedian`	`Gridbox median`
	`Median of the selected grid boxes.`

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`gridcell` *Grid cell quantities*

Description

This module reads the grid cell area of the first grid from the input stream. If the grid cell area is missing it will be computed from the grid coordinates. The area of a grid cell is calculated using spherical triangles from the coordinates of the center and the vertices. The base is a unit sphere which is scaled with the radius of the planet. The default planet radius is 6371000 meter. The parameter radius or the environment variable PLANET_RADIUS can be used to change the default. Depending on the chosen operator the grid cell area or weights are written to the output stream.

Usage

```
cdo_gridarea(ifile, radius = NULL, ofile = NULL)

cdo_gridweights(ifile, radius = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`radius`	FLOAT - Planet radius in meter
`ofile`	String with the path to the output file.

Details

```
gridarea     Grid cell area
        Writes the grid cell area to the output stream. If the grid cell area have to
             be computed it is scaled with the planet radius to square meters.
gridweights  Grid cell weights
             Writes the grid cell area weights to the output stream.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`healpix` *Change healpix resolution*

Description

Degrade or upgrade the resolution of a healpix grid.

Usage

```
cdo_hpdegrade(ifile, nside = NULL, order = NULL, power = NULL, ofile = NULL)

cdo_hpupgrade(ifile, nside = NULL, order = NULL, power = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`nside`	INTEGER - The nside of the target healpix, must be a power of two [default: same as input].
`order`	STRING - Pixel ordering of the target healpix ('nested' or 'ring').
`power`	FLOAT - If non-zero, divide the result by (nside[in]/nside[out])**power. power=-2 keeps the sum of the map invariant.
`ofile`	String with the path to the output file.

Details

```
hpdegrade  Degrade healpix
       Degrade the resolution of a healpix grid. The value of the target pixel is the mean of the source
hpupgrade  Upgrade healpix
       Upgrade the resolution of a healpix grid. The values of the target pixels is the value of the sour
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`histogram` *Histogram*

Description

This module creates bins for a histogram of the input data. The bins have to be adjacent and have non-overlapping intervals. The user has to define the bounds of the bins. The first value is the lower bound and the second value the upper bound of the first bin. The bounds of the second bin are defined by the second and third value, aso. Only 2-dimensional input fields are allowed. The output file contains one vertical level for each of the bins requested.

Usage

```
cdo_histcount(ifile, bounds = NULL, ofile = NULL)

cdo_histfreq(ifile, bounds = NULL, ofile = NULL)

cdo_histmean(ifile, bounds = NULL, ofile = NULL)

cdo_histsum(ifile, bounds = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`bounds`	FLOAT - Comma-separated list of the bin bounds (-inf and inf valid)
`ofile`	String with the path to the output file.

Details

```
histcount  Histogram count
           Number of elements in the bin range.
histsum    Histogram sum
           Sum of elements in the bin range.
histmean   Histogram mean
```

	Mean of elements in the bin range.
`histfreq`	Histogram frequency Relative frequency of elements in the bin range.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`hourpctl` *Hourly percentile values*

Description

This operator computes percentiles over all timesteps of the same hour in infile1. The algorithm uses histograms with minimum and maximum bounds given in infile2 and infile3, respectively. The default number of histogram bins is 101. The default can be overridden by defining the environment variable CDO_PCTL_NBINS. The files infile2 and infile3 should be the result of corresponding hourmin and hourmax operations, respectively. The time of outfile is determined by the time in the middle of all contributing timesteps of infile1. This can be change with the CDO option –timestat_date <first|middle|last>. For every adjacent sequence t_1, ...,t_n of timesteps of the same hour it is: o(t,x) = pth percentile {i(t',x), t_1<t'<=t_n}

Usage

```
cdo_hourpctl(ifile1, ifile2, ifile3, p = NULL, ofile = NULL)
```

Arguments

`ifile1, ifile2, ifile3`	Strings with the path to the input files.
`p`	FLOAT - Percentile number in {0, ..., 100}
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

hourstat *Hourly statistics*

Description

This module computes statistical values over timesteps of the same hour. Depending on the chosen operator the minimum, maximum, range, sum, average, variance or standard deviation of timesteps of the same hour is written to outfile. The time of outfile is determined by the time in the middle of all contributing timesteps of infile. This can be change with the CDO option –timestat_date <first|middle|last>.

Usage

```
cdo_houravg(ifile, ofile = NULL)

cdo_hourmax(ifile, ofile = NULL)

cdo_hourmean(ifile, ofile = NULL)

cdo_hourmin(ifile, ofile = NULL)

cdo_hourrange(ifile, ofile = NULL)

cdo_hourstd(ifile, ofile = NULL)

cdo_hourstd1(ifile, ofile = NULL)

cdo_hoursum(ifile, ofile = NULL)

cdo_hourvar(ifile, ofile = NULL)

cdo_hourvar1(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Details

```
hourmin    Hourly minimum
        For every adjacent sequence t_1, ...,t_n of timesteps of the same hour it is:

            o(t,x) = min\{i(t',x), t_1&lt;t'&lt;=t_n\}
hourmax    Hourly maximum
        For every adjacent sequence t_1, ...,t_n of timesteps of the same hour it is:
```

`o(t,x) = max\{i(t',x), t_1&lt;t'&lt;=t_n\}`

`hourrange` `Hourly range`

For every adjacent sequence `t_1, ...,t_n` of timesteps of the same hour it is:

`o(t,x) = range\{i(t',x), t_1&lt;t'&lt;=t_n\}`

`hoursum` `Hourly sum`

For every adjacent sequence `t_1, ...,t_n` of timesteps of the same hour it is:

`o(t,x) = sum\{i(t',x), t_1&lt;t'&lt;=t_n\}`

`hourmean` `Hourly mean`

For every adjacent sequence `t_1, ...,t_n` of timesteps of the same hour it is:

`o(t,x) = mean\{i(t',x), t_1&lt;t'&lt;=t_n\}`

`houravg` `Hourly average`

For every adjacent sequence `t_1, ...,t_n` of timesteps of the same hour it is:

`o(t,x) = avg\{i(t',x), t_1&lt;t'&lt;=t_n\}`

`hourstd` `Hourly standard deviation`

Normalize by `n`. For every adjacent sequence `t_1, ...,t_n` of timesteps of the same hour it is:

`o(t,x) = std\{i(t',x), t_1&lt;t'&lt;=t_n\}`

`hourstd1` `Hourly standard deviation (n-1)`

Normalize by `(n-1)`. For every adjacent sequence `t_1, ...,t_n` of timesteps of the same hour it is:

`o(t,x) = std1\{i(t',x), t_1&lt;t'&lt;=t_n\}`

`hourvar` `Hourly variance`

Normalize by `n`. For every adjacent sequence `t_1, ...,t_n` of timesteps of the same hour it is:

`o(t,x) = var\{i(t',x), t_1&lt;t'&lt;=t_n\}`

`hourvar1` `Hourly variance (n-1)`

Normalize by `(n-1)`. For every adjacent sequence `t_1, ...,t_n` of timesteps of the same hour it is:

`o(t,x) = var1\{i(t',x), t_1&lt;t'&lt;=t_n\}`

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don’t return filenames return a character vector with the string output.

`hurr` *Hurricane days index per time period*

Description

Let infile be a time series of the daily maximum horizontal wind speed VX, then the number of days where VX is greater than or equal to 32.5 m/s is counted. A further output variable is the maximum number of consecutive days with maximum wind speed greater than or equal to 32.5 m/s. Note that VX is defined as the square root of the sum of squares of the zonal and meridional wind speeds and have to be given in units of m/s. The date information of a timestep in outfile is the date of the last contributing timestep in infile.

Usage

```
cdo_hurr(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`importamsr` *Import AMSR binary files*

Description

This operator imports gridded binary AMSR (Advanced Microwave Scanning Radiometer) data. The binary data files are available from the AMSR ftp site (ftp://ftp.ssmi.com/amsre). Each file consists of twelve (daily) or five (averaged) 0.25 x 0.25 degree grid (1440,720) byte maps. For daily files, six daytime maps in the following order, Time (UTC), Sea Surface Temperature (SST), 10 meter Surface Wind Speed (WSPD), Atmospheric Water Vapor (VAPOR), Cloud Liquid Water (CLOUD), and Rain Rate (RAIN), are followed by six nighttime maps in the same order. Time-Averaged files contain just the geophysical layers in the same order [SST, WSPD, VAPOR, CLOUD, RAIN]. More information to the data is available on the AMSR homepage http://www.remss.com/amsr.

Usage

```
cdo_import_amsr(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`importbinary` *Import binary data sets*

Description

This operator imports gridded binary data sets via a GrADS data descriptor file. The GrADS data descriptor file contains a complete description of the binary data as well as instructions on where to find the data and how to read it. The descriptor file is an ASCII file that can be created easily with a text editor. The general contents of a gridded data descriptor file are as follows: - Filename for the binary data - Missing or undefined data value - Mapping between grid coordinates and world coordinates - Description of variables in the binary data set A detailed description of the components of a GrADS data descriptor file can be found in GrADS. Here is a list of the supported components: BYTESWAPPED, CHSUB, DSET, ENDVARS, FILEHEADER, HEADERBYTES, OPTIONS, TDEF, TITLE, TRAILERBYTES, UNDEF, VARS, XDEF, XYHEADER, YDEF, ZDEF

Usage

```
cdo_import_binary(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

Note

Only 32-bit IEEE floats are supported for standard binary files!

importcmsaf *Import CMSAF HDF5 files*

Description

This operator imports gridded CM-SAF (Satellite Application Facility on Climate Monitoring) HDF5 files. CM-SAF exploits data from polar-orbiting and geostationary satellites in order to provide climate monitoring products of the following parameters: Cloud parameters: cloud fraction (CFC), cloud type (CTY), cloud phase (CPH), cloud top height, pressure and temperature (CTH,CTP,CTT), cloud optical thickness (COT), cloud water path (CWP). Surface radiation components: Surface albedo (SAL); surface incoming (SIS) and net (SNS) shortwave radiation; surface downward (SDL) and outgoing (SOL) longwave radiation, surface net longwave radiation (SNL) and surface radiation budget (SRB). Top-of-atmosphere radiation components: Incoming (TIS) and reflected (TRS) solar radiative flux at top-of-atmosphere. Emitted thermal radiative flux at top-of-atmosphere (TET). Water vapour: Vertically integrated water vapour (HTW), layered vertically integrated water vapour and layer mean temperature and relative humidity for 5 layers (HLW), temperature and mixing ratio at 6 pressure levels. Daily and monthly mean products can be ordered via the CM-SAF web page (www.cmsaf.eu). Products with higher spatial and temporal resolution, i.e. instantaneous swath-based products, are available on request (contact.cmsaf@dwd.de). All products are distributed free-of-charge. More information on the data is available on the CM-SAF homepage (www.cmsaf.eu). Daily and monthly mean products are provided in equal-area projections. CDO reads the projection parameters from the metadata in the HDF5-headers in order to allow spatial operations like remapping. For spatial operations with instantaneous products on original satellite projection, additional files with arrays of latitudes and longitudes are needed. These can be obtained from CM-SAF together with the data.

Usage

```
cdo_import_cmsaf(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

Note

To use this operator, it is necessary to build CDO with HDF5 support (version 1.6 or higher). The PROJ library (version 5.0 or higher) is needed for full support of the remapping functionality.

info — *Information and simple statistics*

Description

This module writes information about the structure and contents for each field of all input files to standard output. A field is a horizontal layer of a data variable. All input files need to have the same structure with the same variables on different timesteps. The information displayed depends on the chosen operator.

Usage

```
cdo_cinfo(ifiles)

cdo_info(ifiles)

cdo_infon(ifiles)

cdo_map(ifiles)
```

Arguments

`ifiles`	Character vector with the path to the input files.

Details

```
info   Dataset information listed by parameter identifier
     Prints information and simple statistics for each field of all input datasets.
       For each field the operator prints one line with the following elements:
       - Date and Time
       - Level, Gridsize and number of Missing values
       - Minimum, Mean and Maximum \\
       The mean value is computed without the use of area weights!
       - Parameter identifier
infon  Dataset information listed by parameter name
       The same as operator info but using the name instead of the
       identifier to label the parameter.
cinfo  Compact information listed by parameter name
     cinfo is a compact version of infon. It prints the minimum, mean and maximum value for each variable
map    Dataset information and simple map
      Prints information, simple statistics and a map for each field of all input
      datasets. The map will be printed only for fields on a regular lon/lat grid.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`input` *Formatted input*

Description

This module reads time series of one 2D variable from standard input. All input fields need to have the same horizontal grid. The format of the input depends on the chosen operator.

Usage

```
cdo_input(grid = NULL, zaxis = NULL, ofile = NULL)

cdo_inputext(grid = NULL, zaxis = NULL, ofile = NULL)

cdo_inputsrv(grid = NULL, zaxis = NULL, ofile = NULL)
```

Arguments

`grid`	STRING - Grid description file or name
`zaxis`	STRING - Z-axis description file
`ofile`	String with the path to the output file.

Details

```
input     ASCII input
          Reads fields with ASCII numbers from standard input and stores them
          in outfile. The numbers read are exactly that ones which are written
          out by the output operator.
inputsrv  SERVICE ASCII input
          Reads fields with ASCII numbers from standard input and stores them
        in outfile. Each field should have a header of 8 integers (SERVICE likely).
         The numbers that are read are exactly that ones which are written out by
          the outputsrv operator.
inputext  EXTRA ASCII input
          Read fields with ASCII numbers from standard input and stores them
         in outfile. Each field should have header of 4 integers (EXTRA likely).
          The numbers read are exactly that ones which are written out by
          the outputext operator.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

intlevel *Linear level interpolation*

Description

This operator performs a linear vertical interpolation of 3D variables. The 1D target levels can be specified with the level parameter or read in via a Z-axis description file.

Usage

```
cdo_intlevel(
  ifile,
  level = NULL,
  zdescription = NULL,
  zvarname = NULL,
  extrapolate = NULL,
  ofile = NULL
)
```

Arguments

ifile	String with the path to the input file.
level	FLOAT - Comma-separated list of target levels
zdescription	STRING - Path to a file containing a description of the Z-axis
zvarname	STRING - Use zvarname as the vertical 3D source coordinate instead of the 1D coordinate variable
extrapolate	BOOL - Fill target layers out of the source layer range with the nearest source layer
ofile	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

intlevel3d *Linear level interpolation from/to 3D vertical coordinates*

Description

This operator performs a linear vertical interpolation of 3D variables fields with given 3D vertical coordinates. infile1 contains the 3D data variables and infile2 the 3D vertical source coordinate. The parameter tgtcoordinate is a datafile with the 3D vertical target coordinate.

Usage

```
cdo_intlevel3d(ifile1, ifile2, tgtcoordinate = NULL, ofile = NULL)

cdo_intlevelx3d(ifile1, ifile2, tgtcoordinate = NULL, ofile = NULL)
```

Arguments

`ifile1`, `ifile2`	Strings with the path to the input files.
`tgtcoordinate`	STRING - filename for 3D vertical target coordinates
`ofile`	String with the path to the output file.

Details

```
intlevel3d   Linear level interpolation onto a 3D vertical coordinate
intlevelx3d  like intlevel3d but with extrapolation
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

inttime *Time interpolation*

Description

This module performs linear interpolation between timesteps. Interpolation is only performed if both values exist. If both values are missing values, the result is also a missing value. If only one value exists, it is taken if the time weighting is greater than or equal to 0.5. So no new value will be created at existing time steps, if the value is missing there.

Usage

```
cdo_intntime(
  ifile,
  date = NULL,
  time = NULL,
  inc = NULL,
  n = NULL,
  ofile = NULL
)

cdo_inttime(
  ifile,
  date = NULL,
  time = NULL,
  inc = NULL,
  n = NULL,
  ofile = NULL
)
```

Arguments

`ifile`	String with the path to the input file.
`date`	STRING - Start date (format YYYY-MM-DD)
`time`	STRING - Start time (format hh:mm:ss)
`inc`	STRING - Optional increment (seconds, minutes, hours, days, months, years) [default: 0hour]
`n`	INTEGER - Number of timesteps from one timestep to the next
`ofile`	String with the path to the output file.

Details

```
inttime   Interpolation between timesteps
      This operator creates a new dataset by linear interpolation between timesteps.
         The user has to define the start date/time with an optional increment.
intntime  Interpolation between timesteps
          This operator performs linear interpolation between timesteps.
      The user has to define the number of timesteps from one timestep to the next.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don’t return filenames return a character vector with the string output.

`intyear` *Year interpolation*

Description

This operator performs linear interpolation between two years, timestep by timestep. The input files need to have the same structure with the same variables. The output files will be named <obase><yyyy><suffix> where yyyy will be the year and suffix is the filename extension derived from the file format.

Usage

```
cdo_intyear(ifile1, ifile2, years = NULL, obase = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`years`	INTEGER - Comma-separated list or first/last[/inc] range of years
`obase`	String with the basename of the output files.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

Note

This operator needs to open all output files simultaneously. The maximum number of open files depends on the operating system!

`invert` *Invert latitudes*

Description

This operator inverts the latitudes of all fields on a rectilinear grid.

Usage

```
cdo_invertlat(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`invertlev` *Invert levels*

Description

This operator inverts the levels of all 3D variables.

Usage

```
cdo_invertlev(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`maggraph` *Line graph plot*

Description

This operator generates line graph plots. The data for the plot is read from infiles. The result is written to outfile. The default output file format is postscript, this can be changed with the device parameter. Here is a list of all graph plot parameters: Keyname & Type & Description device & STRING & Output device (ps, eps, pdf, png, gif, gif_animation, jpeg, svg, kml) ymin & FLOAT & Minimum value of the y-axis data ymax & FLOAT & Maximum value of the y-axis data linewidth & INT & Linewidth (default 8) stat & STRING & "TRUE" or "FALSE to switch on the mean computation. Default is "FALSE". & & Will be overridden to "FALSE if input files have unequal number of time & & steps or different start/end times. sigma & FLOAT & Standard deviation value for generating shaded back ground around the mean value. & & To be used in conjunction with 'stat="TRUE"' obsv & STRING & To indicate if the input files have an observation data, by setting to "TRUE". & & Default value is "FALSE". The observation data should be the first file in the & & input file list. The observation data is always plotted in black colour.

Usage

```
cdo_graph(ifiles, parameter = NULL, ofile = NULL)
```

Arguments

`ifiles`	Character vector with the path to the input files.
`parameter`	STRING - Comma-separated list of plot parameters
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`magplot` *Lat/Lon plot*

Description

The operators in this module generates 2D Lon/Lat plots. The data for the plot is read from infile. Only data on rectilinear Lon/Lat grids are supported. The output file will be named <obase>_<param>.<device> where param is the parameter name and device is the device name. The default output file format is postscript, this can be changed with the device parameter. The type of the plot depends on the choosen operator. Here is a list of all common plot parameters: Keyname & Type & Description device & STRING & Output device (ps, eps, pdf, png, gif, gif_animation, jpeg, svg, kml) projection & STRING & Projection (cylindrical, polar_stereographic, robinson, mercator) style & STRING & Contour line style (solid, dash, dot, chain_dash, chain_dot) min & FLOAT & Minimum value max & FLOAT & Maximum value lon_max & FLOAT & Maximum longitude of the image lon_min & FLOAT & Minimum longitude of the image lat_max & FLOAT & Maximum latitude of the image lat_min & FLOAT & Minimum latitude of the image count & INTEGER & Number of Contour levels / Colour bands interval & FLOAT & Interval in data units between two bands lines list & INTEGER & List of levels to be plotted RGB & STRING & TRUE or FALSE, to indicate, if the input colour is in RGB format step_freq & INTEGER & Frequency of time steps to be considered for making the animation & & (device=gif_animation). Default value is "1" (all time steps). & & Will be ignored if input file has multiple variables. file_split & STRING & TRUE or FALSE, to split the output file for each variable, if input has & & multiple variables. Default value is "FALSE". Valid only for "PS" format.

Usage

```
cdo_contour(ifile, parameter = NULL, ofile = NULL)

cdo_grfill(ifile, parameter = NULL, ofile = NULL)

cdo_shaded(ifile, parameter = NULL, ofile = NULL)
```

Arguments

ifile	String with the path to the input file.
parameter	STRING - Comma-separated list of plot parameters
ofile	String with the path to the output file.

Details

```
contour  Contour plot
      The operator contour generates the discrete contour lines of the input field values.
         The following additional parameters are valid for contour operator,
         module in addition to the common plot parameters:

          Keyname      &amp; Type    &amp; Description
          colour       &amp; STRING  &amp; Colour for drawing the contours
          thickness    &amp; FLOAT   &amp; Thickness of the contour line
      style     &amp; STRING &amp; Line Style can be \&quot;SOLID\ \&quot;DASH\ \&quot;DOT\ \&quot;CH
                       &amp;         &amp; \&quot;CHAIN_DOT\&quot;
shaded   Shaded contour plot
      The operator shaded generates the filled contours of the given input field values.
      The following additional parameters are valid for shaded contour and gridfill operator,
         in addition to the common plot parameters.

          Keyname      &amp; Type    &amp; Description
          colour_min   &amp; STRING  &amp; Colour for the Minimum colour band
          colour_max   &amp; STRING  &amp; Colour for the Minimum colour band
      colour_triad &amp; STRING  &amp; Direction of colour sequencing for shading \&quot;CW\&quot; or \
            &amp;      &amp; to denote \&quot;clockwise\&quot; and \&quot;anticlockwise\&quot; res
            &amp;      &amp; To be used in conjunction with \&quot;colour_min\ \&quot;colour_max\&q
                   &amp;         &amp; options. Default is \&quot;ACW\&quot;
      colour_table &amp; STRING  &amp; File with user specified colours with the format as

         Example file for 6 colours in RGB format:
          6
          RGB(0.0;0.0;1.0)
          RGB(0.0;0.0;0.5)
          RGB(0.0;0.5;0.5)
          RGB(0.0;1.0;0.0)
          RGB(0.5;0.5;0.0)
          RGB(1.0;0.0;0.0)

grfill   Shaded gridfill plot
      The operator grfill is similar to satellite imaging and shades each cell (pixel) according
         to the value of the field at that cell.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

Note

All colour parameter can be either standard name or in RGB format. The valid standard name strings for \"colour\" are: \"red\ \"green\ \"blue\ \"yellow\ \"cyan\ \"magenta\ \"black\ \"avocado\ \"beige\ \"brick\ \"brown\ \"burgundy\ \"charcoal\ \"chestnut\ \"coral\ \"cream\ \"evergreen\ \"gold\ \"grey\ \"khaki\ \"kellygreen\ \"lavender\ \"mustard\ \"navy\ \"ochre\ \"olive\ \"peach\ \"pink\ \"rose\ \"rust\ \"sky\ \"tan\ \"tangerine\ \"turquoise\ \"violet\ \"reddishpurple\ \"purplered\ \"purplishred\ \"orangishred\ \"redorange\ \"reddishorange\ \"orange\ \"yellowishorange\ \"orangeyellow\ \"orangishyellow\ \"greenishyellow\ \"yellowgreen\ \"yellowishgreen\ \"bluishgreen\ \"bluegreen\ \"greenishblue\ \"purplishblue\ \"bluepurple\ \"bluishpurple\ \"purple\ \"white\"

`magvector` *Lat/Lon vector plot*

Description

This operator generates 2D Lon/Lat vector plots. The data for the plot is read from infile. The input is expected to contain two velocity components. Only data on rectilinear Lon/Lat grids are supported. The output file will be named <obase>.<device> where device is the device name. The default output file format is postscript, this can be changed with the device parameter. Here is a list of all vector plot parameters: Keyname & Type & Description device & STRING & Output device (ps, eps, pdf, png, gif, gif_animation, jpeg, svg, kml) projection & STRING & Projection (cylindrical, polar_stereographic, robinson, mercator) thin_fac & FLOAT & Controls the actual number of wind arrows or flags plotted (default 2). unit_vec & FLOAT & Wind speed in m/s represented by a unit vector (1.0cm) step_freq & INTEGER & Frequency of time steps to be considered for making the animation & & (device=gif_animation). Default value is "1" (all time steps). & & Will be ignored if input file has multiple variables.

Usage

```
cdo_vector(ifile, parameter = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`parameter`	STRING - Comma-separated list of plot parameters
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`mapreduce` *Reduce fields to userdefined mask*

Description

This module holds an operator for data reduction based on a user defined mask. The output grid is unstructured and includes coordinate bounds. Bounds can be avoided by using the additional ’nobounds’ keyword. With ’nocoords’ given, coordinates a completely suppressed.

Usage

```
cdo_reducegrid(ifile, mask = NULL, limitCoordsOutput = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`mask`	STRING - file which holds the mask field
`limitCoordsOutput`	STRING - optional parameter to limit coordinates output: ’nobounds’ disables coordinate bounds, ’nocoords’ avoids all coordinate information
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don’t return filenames return a character vector with the string output.

`maskbox` *Mask a box*

Description

Masks grid cells inside a lon/lat or index box. The elements inside the box are untouched, the elements outside are set to missing value. All input fields need to have the same horizontal grid. Use sellonlatbox or selindexbox if only the data inside the box are needed.

Usage

```
cdo_maskindexbox(
  ifile,
  lon1 = NULL,
  lon2 = NULL,
  lat1 = NULL,
  lat2 = NULL,
  idx1 = NULL,
  idx2 = NULL,
  idy1 = NULL,
  idy2 = NULL,
  ofile = NULL
)

cdo_masklonlatbox(
  ifile,
  lon1 = NULL,
  lon2 = NULL,
  lat1 = NULL,
  lat2 = NULL,
  idx1 = NULL,
  idx2 = NULL,
  idy1 = NULL,
  idy2 = NULL,
  ofile = NULL
)
```

Arguments

`ifile`	String with the path to the input file.
`lon1`	FLOAT - Western longitude
`lon2`	FLOAT - Eastern longitude
`lat1`	FLOAT - Southern or northern latitude
`lat2`	FLOAT - Northern or southern latitude
`idx1`	INTEGER - Index of first longitude
`idx2`	INTEGER - Index of last longitude
`idy1`	INTEGER - Index of first latitude
`idy2`	INTEGER - Index of last latitude
`ofile`	String with the path to the output file.

Details

```
masklonlatbox  Mask a longitude/latitude box
          Masks grid cells inside a lon/lat box. The user must specify the longitude and latitude of the
          Only those grid cells are considered whose grid center lies within the lon/lat box.
          For rotated lon/lat grids the parameters must be specified in rotated coordinates.
```

`maskindexbox` `Mask an index box`

`Masks grid cells within an index box. The user must specify the indices of the edges of the box`
`The index of the left edge can be greater then the one of the right edge. Use negative indexing`
`start from the end. The input grid must be a regular lon/lat or a 2D curvilinear grid.`

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`maskregion` *Mask regions*

Description

Masks different regions of the input fields. The grid cells inside a region are untouched, the cells outside are set to missing value. Considered are only those grid cells with the grid center inside the regions. All input fields must have the same horizontal grid. Regions can be defined by the user via an ASCII file. Each region consists of the geographic coordinates of a polygon. Each line of a polygon description file contains the longitude and latitude of one point. Each polygon description file can contain one or more polygons separated by a line with the character &. Predefined regions of countries can be specified via the country codes. A country is specified with dcw:<CountryCode>. Country codes can be combined with the plus sign.

Usage

```
cdo_maskregion(ifile, regions = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`regions`	STRING - Comma-separated list of ASCII formatted files with different regions
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`mastrfu` *Mass stream function*

Description

This is a special operator for the post processing of the atmospheric general circulation model ECHAM. It computes the mass stream function (code=272). The input dataset have to be a zonal mean of v-velocity [m/s] (code=132) on pressure levels.

Usage

```
cdo_mastrfu(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`math` *Mathematical functions*

Description

This module contains some standard mathematical functions. All trigonometric functions calculate with radians.

Usage

```
cdo_abs(ifile, ofile = NULL)

cdo_acos(ifile, ofile = NULL)

cdo_asin(ifile, ofile = NULL)

cdo_atan(ifile, ofile = NULL)

cdo_cos(ifile, ofile = NULL)

cdo_exp(ifile, ofile = NULL)
```

```
cdo_int(ifile, ofile = NULL)

cdo_ln(ifile, ofile = NULL)

cdo_log10(ifile, ofile = NULL)

cdo_nint(ifile, ofile = NULL)

cdo_not(ifile, ofile = NULL)

cdo_pow(ifile, ofile = NULL)

cdo_reci(ifile, ofile = NULL)

cdo_sin(ifile, ofile = NULL)

cdo_sqr(ifile, ofile = NULL)

cdo_sqrt(ifile, ofile = NULL)

cdo_tan(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Details

```
abs    Absolute value
       o(t,x) = abs(i(t,x))
int    Integer value
       o(t,x) = int(i(t,x))
nint   Nearest integer value
       o(t,x) = nint(i(t,x))
pow    Power
       o(t,x) = i(t,x)^y
sqr    Square
       o(t,x) = i(t,x)^2
sqrt   Square root
       o(t,x) = sqrt(i(t,x))
exp    Exponential
       o(t,x) = e^i(t,x)
ln     Natural logarithm
       o(t,x) = ln(i(t,x))
log10  Base 10 logarithm
       o(t,x) = log10(i(t,x))
sin    Sine
```

```
         o(t,x) = sin(i(t,x))
cos      Cosine
         o(t,x) = cos(i(t,x))
tan      Tangent
         o(t,x) = tan(i(t,x))
asin     Arc sine
         o(t,x) = asin(i(t,x))
acos     Arc cosine
         o(t,x) = acos(i(t,x))
atan     Arc tangent
         o(t,x) = atan(i(t,x))
reci     Reciprocal value
         o(t,x) = 1 / i(t,x)
not      Logical NOT
         o(t,x) = 1, if x equal 0; else 0
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

---

merge *Merge datasets*

Description

This module reads datasets from several input files, merges them and writes the resulting dataset to outfile.

Usage

```
cdo_merge(ifiles, skip_same_time = NULL, names = NULL, ofile = NULL)

cdo_mergetime(ifiles, skip_same_time = NULL, names = NULL, ofile = NULL)
```

Arguments

`ifiles`	Character vector with the path to the input files.
`skip_same_time`	BOOL - Skips all consecutive timesteps with a double entry of the same timestamp.
`names`	STRING - Fill missing variable names with missing values (union) or use the intersection (intersect).
`ofile`	String with the path to the output file.

Details

`merge`	`Merge datasets with different fields` `Merges time series of different fields from several input datasets. The number of fields per timestep written to outfile is the sum of the field numbers per timestep in all input datasets. The time series on all input datasets are required to have different fields and the same number of timesteps. The fields in each different input file either have to be different variables or different levels of the same variable. A mixture of different variables on different levels in different input files is not allowed.`
`mergetime`	`Merge datasets sorted by date and time` `Merges all timesteps of all input files sorted by date and time. All input files need to have the same structure with the same variables on different timesteps. After this operation every input timestep is in outfile and all timesteps are sorted by date and time.`

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

Note

Operators of this module need to open all input files simultaneously. The maximum number of open files depends on the operating system!

`mergegrid` *Merge grid*

Description

Merges grid points of all variables from infile2 to infile1 and write the result to outfile. Only the non missing values of infile2 will be used. The horizontal grid of infile2 should be smaller or equal to the grid of infile1 and the resolution must be the same. Only rectilinear grids are supported. Both input files need to have the same variables and the same number of timesteps.

Usage

```
cdo_mergegrid(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`merstat` *Meridional statistics*

Description

This module computes meridional statistical values of the input fields. Depending on the chosen operator, the meridional minimum, maximum, range, sum, average, standard deviation, variance, skewness, kurtosis, median or a certain percentile of the field is written to outfile. Operators of this module require all variables on the same regular lon/lat grid.

Usage

```
cdo_meravg(ifile, p = NULL, ofile = NULL)

cdo_merkurt(ifile, p = NULL, ofile = NULL)

cdo_mermax(ifile, p = NULL, ofile = NULL)

cdo_mermean(ifile, p = NULL, ofile = NULL)

cdo_mermedian(ifile, p = NULL, ofile = NULL)

cdo_mermin(ifile, p = NULL, ofile = NULL)

cdo_merpctl(ifile, p = NULL, ofile = NULL)

cdo_merrange(ifile, p = NULL, ofile = NULL)

cdo_merskew(ifile, p = NULL, ofile = NULL)

cdo_merstd(ifile, p = NULL, ofile = NULL)

cdo_merstd1(ifile, p = NULL, ofile = NULL)

cdo_mersum(ifile, p = NULL, ofile = NULL)

cdo_mervar(ifile, p = NULL, ofile = NULL)

cdo_mervar1(ifile, p = NULL, ofile = NULL)
```

Arguments

ifile	String with the path to the input file.
p	FLOAT - Percentile number in {0, ..., 100}
ofile	String with the path to the output file.

Details

```
mermin     Meridional minimum
           For every longitude the minimum over all latitudes is computed.
mermax     Meridional maximum
           For every longitude the maximum over all latitudes is computed.
merrange   Meridional range
           For every longitude the range over all latitudes is computed.
mersum     Meridional sum
           For every longitude the sum over all latitudes is computed.
mermean    Meridional mean
          For every longitude the area weighted mean over all latitudes is computed.
meravg     Meridional average
       For every longitude the area weighted average over all latitudes is computed.
merstd     Meridional standard deviation
       For every longitude the standard deviation over all latitudes is computed. Normalize by n.
merstd1    Meridional standard deviation (n-1)
       For every longitude the standard deviation over all latitudes is computed. Normalize by (n-1).
mervar     Meridional variance
       For every longitude the variance over all latitudes is computed. Normalize by n.
mervar1    Meridional variance (n-1)
       For every longitude the variance over all latitudes is computed. Normalize by (n-1).
merskew    Meridional skewness
           For every longitude the skewness over all latitudes is computed.
merkurt    Meridional kurtosis
           For every longitude the kurtosis over all latitudes is computed.
mermedian  Meridional median
           For every longitude the median over all latitudes is computed.
merpctl    Meridional percentiles
          For every longitude the pth percentile over all latitudes is computed.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`monarith` *Monthly arithmetic*

Description

This module performs simple arithmetic of a time series and one timestep with the same month and year. For each field in infile1 the corresponding field of the timestep in infile2 with the same month and year is used. The input files need to have the same structure with the same variables. Usually infile2 is generated by an operator of the module MONSTAT.

Usage

```
cdo_monadd(ifile1, ifile2, ofile = NULL)

cdo_mondiv(ifile1, ifile2, ofile = NULL)

cdo_monmul(ifile1, ifile2, ofile = NULL)

cdo_monsub(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Details

```
monadd  Add monthly time series
        Adds a time series and a monthly time series.
monsub  Subtract monthly time series
        Subtracts a time series and a monthly time series.
monmul  Multiply monthly time series
        Multiplies a time series and a monthly time series.
mondiv  Divide monthly time series
        Divides a time series and a monthly time series.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

monpctl *Monthly percentile values*

Description

This operator computes percentiles over all timesteps of the same month in infile1. The algorithm uses histograms with minimum and maximum bounds given in infile2 and infile3, respectively. The default number of histogram bins is 101. The default can be overridden by defining the environment variable CDO_PCTL_NBINS. The files infile2 and infile3 should be the result of corresponding monmin and monmax operations, respectively. The time of outfile is determined by the time in the middle of all contributing timesteps of infile1. This can be change with the CDO option –timestat_date <first|middle|last>. For every adjacent sequence t_1, ...,t_n of timesteps of the same month it is: o(t,x) = pth percentile {i(t',x), t_1<t'<=t_n}

Usage

```
cdo_monpctl(ifile1, ifile2, ifile3, p = NULL, ofile = NULL)
```

Arguments

`ifile1, ifile2, ifile3`	Strings with the path to the input files.
`p`	FLOAT - Percentile number in {0, ..., 100}
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

monstat *Monthly statistics*

Description

This module computes statistical values over timesteps of the same month. Depending on the chosen operator the minimum, maximum, range, sum, average, variance or standard deviation of timesteps of the same month is written to outfile. The time of outfile is determined by the time in the middle of all contributing timesteps of infile. This can be change with the CDO option –timestat_date <first|middle|last>.

Usage

```
cdo_monavg(ifile, complete_only = NULL, ofile = NULL)

cdo_monmax(ifile, complete_only = NULL, ofile = NULL)

cdo_monmean(ifile, complete_only = NULL, ofile = NULL)

cdo_monmin(ifile, complete_only = NULL, ofile = NULL)

cdo_monrange(ifile, complete_only = NULL, ofile = NULL)

cdo_monstd(ifile, complete_only = NULL, ofile = NULL)

cdo_monstd1(ifile, complete_only = NULL, ofile = NULL)

cdo_monsum(ifile, complete_only = NULL, ofile = NULL)

cdo_monvar(ifile, complete_only = NULL, ofile = NULL)

cdo_monvar1(ifile, complete_only = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`complete_only`	BOOL - Process the last month only if it is complete
`ofile`	String with the path to the output file.

Details

```
monmin    Monthly minimum
      For every adjacent sequence t_1, ...,t_n of timesteps of the same month it is:

          o(t,x) = min\{i(t',x), t_1&lt;t'&lt;=t_n\}
monmax    Monthly maximum
      For every adjacent sequence t_1, ...,t_n of timesteps of the same month it is:

          o(t,x) = max\{i(t',x), t_1&lt;t'&lt;=t_n\}
monrange  Monthly range
      For every adjacent sequence t_1, ...,t_n of timesteps of the same month it is:

          o(t,x) = range\{i(t',x), t_1&lt;t'&lt;=t_n\}
monsum    Monthly sum
      For every adjacent sequence t_1, ...,t_n of timesteps of the same month it is:

          o(t,x) = sum\{i(t',x), t_1&lt;t'&lt;=t_n\}
monmean   Monthly mean
      For every adjacent sequence t_1, ...,t_n of timesteps of the same month it is:
          o(t,x) = mean\{i(t',x), t_1&lt;t'&lt;=t_n\}
monavg    Monthly average
       For every adjacent sequence t_1, ...,t_n of timesteps of the same month it is:

          o(t,x) = avg\{i(t',x), t_1&lt;t'&lt;=t_n\}
monstd    Monthly standard deviation
       Normalize by n. For every adjacent sequence t_1, ...,t_n of timesteps of the same month it is:

          o(t,x) = std\{i(t',x), t_1 &lt; t' &lt;= t_n\}
monstd1   Monthly standard deviation (n-1)
       Normalize by (n-1). For every adjacent sequence t_1, ...,t_n of timesteps of the same month it is:

          o(t,x) = std1\{i(t',x), t_1 &lt; t' &lt;= t_n\}
monvar    Monthly variance
       Normalize by n. For every adjacent sequence t_1, ...,t_n of timesteps of the same month it is:

          o(t,x) = var\{i(t',x), t_1 &lt; t' &lt;= t_n\}
monvar1   Monthly variance (n-1)
       Normalize by (n-1). For every adjacent sequence t_1, ...,t_n of timesteps of the same month it is:

          o(t,x) = var1\{i(t',x), t_1 &lt; t' &lt;= t_n\}
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

---

mrotuvb *Backward rotation of MPIOM data*

Description

MPIOM data are on a rotated Arakawa C grid. The velocity components U and V are located on the edges of the cells and point in the direction of the grid lines and rows. With mrotuvb the velocity vector is rotated in latitudinal and longitudinal direction. Before the rotation, U and V are interpolated to the scalar points (cell center). U is located with the coordinates for U in infile1 and V in infile2. mrotuvb assumes a positive meridional flow for a flow from grid point(i,j) to grid point(i,j+1) and positive zonal flow for a flow from grid point(i+1,j) to point(i,j).

Usage

```
cdo_mrotuvb(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

Note

This is a specific implementation for data from the MPIOM model, it may not work with data from other sources.

`ncl_wind` *Wind transformation*

Description

This module contains CDO operators with an interface to NCL functions. The corresponding NCL functions have the same name. A more detailed description of those NCL function can be found on the NCL homepage https://www.ncl.ucar.edu.

Usage

```
cdo_uv2dv_cfd(
  ifile,
  u = NULL,
  v = NULL,
  boundOpt = NULL,
  outMode = NULL,
  ofile = NULL
)

cdo_uv2vr_cfd(
  ifile,
  u = NULL,
  v = NULL,
  boundOpt = NULL,
  outMode = NULL,
  ofile = NULL
)
```

Arguments

`ifile`	String with the path to the input file.
`u`	STRING - Name of variable u (default: u)
`v`	STRING - Name of variable v (default: v)
`boundOpt`	INTEGER - Boundary condition option (0-3) (default: 0/1 for cyclic grids)
`outMode`	STRING - Output mode new/append (default: new)
`ofile`	String with the path to the output file.

Details

`uv2vr_cfd` U and V wind to relative vorticity
: Computes relative vorticity for a latitude-longitude grid using centered finite differences. The grid need not be global and missing values are allowed.

`uv2dv_cfd` U and V wind to divergence
: Computes divergence for a latitude-longitude grid using centered finite differences. The grid need not be global and missing values are allowed.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`ninfo` *Print the number of parameters, levels or times*

Description

This module prints the number of variables, levels or times of the input dataset.

Usage

```
cdo_ndate(ifile)

cdo_ngridpoints(ifile)

cdo_ngrids(ifile)

cdo_nlevel(ifile)

cdo_nmon(ifile)

cdo_npar(ifile)

cdo_ntime(ifile)

cdo_nyear(ifile)
```

Arguments

`ifile`	String with the path to the input file.

Details

`npar`	`Number of parameters`
	`Prints the number of parameters (variables).`

`nlevel`	`Number of levels`
	`Prints the number of levels for each variable.`

`nyear`	`Number of years`
	`Prints the number of different years.`

`nmon`	`Number of months`
	`Prints the number of different combinations of years and months.`

`ndate`	`Number of dates`
	`Prints the number of different dates.`

`ntime`	`Number of timesteps`
	`Prints the number of timesteps.`

`ngridpoints`	`Number of gridpoints`
	`Prints the number of gridpoints for each variable.`

`ngrids`	`Number of horizontal grids`
	`Prints the number of horizontal grids.`

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`output` *Formatted output*

Description

This module prints all values of all input datasets to standard output. All input fields need to have the same horizontal grid. All input files need to have the same structure with the same variables. The format of the output depends on the chosen operator.

Usage

```
cdo_output(ifiles, format = NULL, nelem = NULL)

cdo_outputext(ifiles, format = NULL, nelem = NULL)

cdo_outputf(ifiles, format = NULL, nelem = NULL)

cdo_outputint(ifiles, format = NULL, nelem = NULL)

cdo_outputsrv(ifiles, format = NULL, nelem = NULL)
```

Arguments

ifiles	Character vector with the path to the input files.
format	STRING - C-style format for one element (e.g. %13.6g)
nelem	INTEGER - Number of elements for each row (default: nelem = 1)

Details

```
output     ASCII output
           Prints all values to standard output.
          Each row has 6 elements with the C-style format \&quot;%13.6g\&quot;.
outputf    Formatted output
           Prints all values to standard output.
       The format and number of elements for each row have to be specified by the parameters
           format and nelem. The default for nelem is 1.
outputint  Integer output
           Prints all values rounded to the nearest integer to standard output.
outputsrv  SERVICE ASCII output
           Prints all values to standard output.
           Each field with a header of 8 integers (SERVICE likely).
outputext  EXTRA ASCII output
           Prints all values to standard output.
           Each field with a header of 4 integers (EXTRA likely).
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

---

outputgmt *GMT output*

Description

This module prints the first field of the input dataset to standard output. The output can be used to generate 2D Lon/Lat plots with GMT. The format of the output depends on the chosen operator.

Usage

```
cdo_gmtcells(ifile)

cdo_gmtxyz(ifile)
```

Arguments

ifile	String with the path to the input file.

Details

```
gmtxyz    GMT xyz format
          The operator exports the first field to the GMT xyz ASCII format.
       The output can be used to create contour plots with the GMT module pscontour.
gmtcells  GMT multiple segment format
       The operator exports the first field to the GMT multiple segment ASCII format.
       The output can be used to create shaded gridfill plots with the GMT module psxy.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`outputtab` *Table output*

Description

This operator prints a table of all input datasets to standard output. infiles is an arbitrary number of input files. All input files need to have the same structure with the same variables on different timesteps. All input fields need to have the same horizontal grid. The contents of the table depends on the chosen parameters. The format of each table parameter is keyname[:len]. len is the optional length of a table entry. The number of significant digits of floating point parameters can be set with the CDO option –precision, the default is 7. Here is a list of all valid keynames: Keyname & Type & Description value & FLOAT & Value of the variable [len:8] name & STRING & Name of the variable [len:8] param & STRING & Parameter ID (GRIB1: code[.tabnum]; GRIB2: num[.cat[.dis]]) [len:11] code & INTEGER & Code number [len:4] x & FLOAT & X coordinate of the original grid [len:6] y & FLOAT & Y coordinate of the original grid [len:6] lon & FLOAT & Longitude coordinate in degrees [len:6] lat & FLOAT & Latitude coordinate in degrees [len:6] lev & FLOAT & Vertical level [len:6] xind & INTEGER & Grid x index [len:4] yind & INTEGER & Grid y index [len:4] timestep & INTEGER & Timestep number [len:6] date & STRING & Date (format YYYY-MM-DD) [len:10] time & STRING & Time (format hh:mm:ss) [len:8] year & INTEGER & Year [len:5] month & INTEGER & Month [len:2] day & INTEGER & Day [len:2] nohead & INTEGER & Disable output of header line

Usage

```
cdo_outputtab(ifiles, parameter = NULL)
```

Arguments

`ifiles`	Character vector with the path to the input files.
`parameter`	STRING - Comma-separated list of keynames, one for each column of the table

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`pack` *Pack data*

Description

Packing reduces the data volume by reducing the precision of the stored numbers. It is implemented using the NetCDF attributes add_offset and scale_factor. The operator pack calculates the attributes add_offset and scale_factor for all variables. The default data type for all variables is automatically changed to 16-bit integer. Use the CDO option -b to change the data type to a different integer precision, if needed. Missing values are automatically transformed to the current data type. Alternatively, the pack parameters add_offset and scale_factor can be read from a file for each variable.

Usage

```
cdo_pack(ifile, printparam = NULL, filename = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`printparam`	BOOL - Print pack parameters to stdout for each variable
`filename`	STRING - Read pack parameters from file for each variable[format: name=<> add_offset=<> scale_factor=<>]
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

pressure *Pressure on model levels*

Description

This module contains operators to calculate the pressure on model levels. To calculate the pressure on model levels, the a and b coefficients defining the model levels and the surface pressure are required. The a and b coefficients are normally part of the model level data. If not available, the surface pressure can be derived from the logarithm of the surface pressure. The surface pressure is identified by the GRIB1 code number or NetCDF CF standard name. Name & Units & GRIB1 code & CF standard name log surface pressure & Pa & 152 & surface pressure & Pa & 134 & surface_air_pressure

Usage

```
cdo_delta_pressure(ifile, ofile = NULL)

cdo_pressure(ifile, ofile = NULL)

cdo_pressure_half(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Details

```
pressure_half   Pressure on half-levels
              This operator computes the pressure on model half-levels in pascal.
                The model half-level pressure (p_half) is given by:


                        p_half = a + b * sp

                with
                   a, b: coefficients defining the model levels
                   sp: surface pressure
pressure        Pressure on full-levels
              This operator computes the pressure on model full-levels in pascal.
           The pressure on model full-levels (p_full) is in the middle of the layers defined by the model

                        p_full = (p_half_above + p_half_below) / 2

delta_pressure  Pressure difference of half-levels
           This operator computes the pressure difference between to model half-levels.

                        delta_p = p_half_below - p_half_above
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`regres` *Regression*

Description

The values of the input file infile are assumed to be distributed as N(a+b*t,S^2) with unknown a, b and S^2. This operator estimates the parameter b. For every field element x only those timesteps t belong to the sample S(x), which have i(t,x) NE miss. It is assumed that all timesteps are equidistant, if this is not the case set the parameter equal=false.

Usage

```
cdo_regres(ifile, equal = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`equal`	BOOL - Set to false for unequal distributed timesteps (default: true)
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`remap` *Grid remapping*

Description

Interpolation between different horizontal grids can be a very time-consuming process. Especially if the data are on an unstructured and/or a large grid. In this case the interpolation process can be split into two parts. Firstly the generation of the interpolation weights, which is the most time-consuming part. These interpolation weights can be reused for every remapping process with the operator remap. This operator remaps all input fields to a new horizontal grid. The remap type and the interpolation weights of one input grid are read from a NetCDF file. More weights are computed if the input fields are on different grids. The NetCDF file with the weights should follow the SCRIP convention. Normally these weights come from a previous call to one of the genXXX operators (e.g. genbil) or were created by the original SCRIP package.

Usage

```
cdo_remap(ifile, grid = NULL, weights = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`grid`	STRING - Target grid description file or name
`weights`	STRING - Interpolation weights (SCRIP NetCDF file)
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`remapbic` *Bicubic interpolation*

Description

This module contains operators for a bicubic remapping of fields between grids in spherical coordinates. The interpolation is based on an adapted SCRIP library version. For a detailed description of the interpolation method see SCRIP. This interpolation method only works on quadrilateral curvilinear source grids.

Usage

```
cdo_genbic(ifile, grid = NULL, map3d = NULL, ofile = NULL)

cdo_remapbic(ifile, grid = NULL, map3d = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`grid`	STRING - Target grid description file or name
`map3d`	BOOL - Generate all mapfiles of the first 3D field
`ofile`	String with the path to the output file.

Details

```
remapbic  Bicubic interpolation
          Performs a bicubic interpolation on all input fields.
genbic    Generate bicubic interpolation weights
      Generates bicubic interpolation weights for the first input field and writes the
      result to a file. The format of this file is NetCDF following the SCRIP convention.
      Use the operator remap to apply this remapping weights to a data file with the same source grid.
      Set the parameter map3d=true to generate all mapfiles of the first 3D field with varying masks.
      In this case the mapfiles will be named &lt;outfile&gt;&lt;xxx&gt;.nc. xxx will have five digits
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

remapbil *Bilinear interpolation*

Description

This module contains operators for a bilinear remapping of fields between grids in spherical coordinates. The interpolation is based on an adapted SCRIP library version. For a detailed description of the interpolation method see SCRIP. This interpolation method only works on quadrilateral curvilinear source grids.

Usage

```
cdo_genbil(ifile, grid = NULL, map3d = NULL, ofile = NULL)

cdo_remapbil(ifile, grid = NULL, map3d = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`grid`	STRING - Target grid description file or name
`map3d`	BOOL - Generate all mapfiles of the first 3D field
`ofile`	String with the path to the output file.

Details

```
remapbil  Bilinear interpolation
          Performs a bilinear interpolation on all input fields.
genbil    Generate bilinear interpolation weights
      Generates bilinear interpolation weights for the first input field and writes the
      result to a file. The format of this file is NetCDF following the SCRIP convention.
      Use the operator remap to apply this remapping weights to a data file with the same source grid.
      Set the parameter map3d=true to generate all mapfiles of the first 3D field with varying masks.
      In this case the mapfiles will be named &lt;outfile&gt;&lt;xxx&gt;.nc. xxx will have five digits
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`remapcon` *First order conservative remapping*

Description

This module contains operators for a first order conservative remapping of fields between grids in spherical coordinates. The operators in this module uses code from the YAC software package to compute the conservative remapping weights. For a detailed description of the interpolation method see YAC. The interpolation method is completely general and can be used for any grid on a sphere. The search algorithm for the conservative remapping requires that no grid cell occurs more than once.

Usage

```
cdo_gencon(ifile, grid = NULL, map3d = NULL, ofile = NULL)

cdo_remapcon(ifile, grid = NULL, map3d = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`grid`	STRING - Target grid description file or name
`map3d`	BOOL - Generate all mapfiles of the first 3D field
`ofile`	String with the path to the output file.

Details

```
remapcon  First order conservative remapping
          Performs a first order conservative remapping on all input fields.
gencon    Generate 1st order conservative remap weights
       Generates first order conservative remapping weights for the first input field and
       writes the result to a file. The format of this file is NetCDF following the SCRIP convention.
       Use the operator remap to apply this remapping weights to a data file with the same source grid.
       Set the parameter map3d=true to generate all mapfiles of the first 3D field with varying masks.
       In this case the mapfiles will be named &lt;outfile&gt;&lt;xxx&gt;.nc. xxx will have five digits
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

---

remapdis *Distance weighted average remapping*

Description

This module contains operators for an inverse distance weighted average remapping of the four nearest neighbor values of fields between grids in spherical coordinates. The default number of 4 neighbors can be changed with the neighbors parameter.

Usage

```
cdo_gendis(ifile, grid = NULL, neighbors = NULL, map3d = NULL, ofile = NULL)

cdo_remapdis(ifile, grid = NULL, neighbors = NULL, map3d = NULL, ofile = NULL)
```

Arguments

ifile	String with the path to the input file.
grid	STRING - Target grid description file or name
neighbors	INTEGER - Number of nearest neighbors [default: 4]
map3d	BOOL - Generate all mapfiles of the first 3D field
ofile	String with the path to the output file.

Details

```
remapdis  Distance weighted average remapping
      Performs an inverse distance weighted averaged remapping of the nearest neighbor values on all in
gendis    Generate distance weighted average remap weights
      Generates distance weighted averaged remapping weights of the nearest neighbor values for the fir
      field and writes the result to a file. The format of this file is NetCDF following the SCRIP conver
      Use the operator remap to apply this remapping weights to a data file with the same source grid.
      Set the parameter map3d=true to generate all mapfiles of the first 3D field with varying masks.
      In this case the mapfiles will be named &lt;outfile&gt;&lt;xxx&gt;.nc. xxx will have five digits
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`remapeta` *Remap vertical hybrid level*

Description

This operator interpolates between different vertical hybrid levels. This include the preparation of consistent data for the free atmosphere. The procedure for the vertical interpolation is based on the HIRLAM scheme and was adapted from INTERA. The vertical interpolation is based on the vertical integration of the hydrostatic equation with few adjustments. The basic tasks are the following one: - at first integration of hydrostatic equation - extrapolation of surface pressure - Planetary Boundary-Layer (PBL) proutfile interpolation - interpolation in free atmosphere - merging of both proutfiles - final surface pressure correction The vertical interpolation corrects the surface pressure. This is simply a cut-off or an addition of air mass. This mass correction should not influence the geostrophic velocity field in the middle troposhere. Therefore the total mass above a given reference level is conserved. As reference level the geopotential height of the 400 hPa level is used. Near the surface the correction can affect the vertical structure of the PBL. Therefore the interpolation is done using the potential temperature. But in the free atmosphere above a certain n (n=0.8 defining the top of the PBL) the interpolation is done linearly. After the interpolation both proutfiles are merged. With the resulting temperature/pressure correction the hydrostatic equation is integrated again and adjusted to the reference level finding the final surface pressure correction. A more detailed description of the interpolation can be found in INTERA. This operator requires all variables on the same horizontal grid.

Usage

```
cdo_remapeta(ifile, vct = NULL, oro = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`vct`	STRING - File name of an ASCII dataset with the vertical coordinate table
`oro`	STRING - File name with the orography (surf. geopotential) of the target dataset (optional)
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

Note

The code numbers or the variable names of the required parameter have to follow the ECHAM convention. Use the sinfo command to test if your vertical coordinate system is recognized as hybrid system. In case remapeta complains about not finding any data on hybrid model levels you may wish to use the setzaxis command to generate a zaxis description which conforms to the ECHAM convention. See section \"1.4 Z-axis description\" for an example how to define a hybrid Z-axis.

`remaplaf` *Largest area fraction remapping*

Description

This module contains operators for a largest area fraction remapping of fields between grids in spherical coordinates. The operators in this module uses code from the YAC software package to compute the largest area fraction. For a detailed description of the interpolation method see YAC. The interpolation method is completely general and can be used for any grid on a sphere. The search algorithm for this remapping method requires that no grid cell occurs more than once.

Usage

```
cdo_genlaf(ifile, grid = NULL, ofile = NULL)

cdo_remaplaf(ifile, grid = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`grid`	STRING - Target grid description file or name
`ofile`	String with the path to the output file.

Details

```
remaplaf  Largest area fraction remapping
          Performs a largest area fraction remapping on all input fields.
genlaf    Generate largest area fraction remap weights
       Generates largest area fraction remapping weights for the first input field and
       writes the result to a file. The format of this file is NetCDF following the SCRIP convention.
       Use the operator remap to apply this remapping weights to a data file with the same source grid.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`remapnn` *Nearest neighbor remapping*

Description

This module contains operators for a nearest neighbor remapping of fields between grids in spherical coordinates.

Usage

```
cdo_gennn(ifile, grid = NULL, map3d = NULL, ofile = NULL)

cdo_remapnn(ifile, grid = NULL, map3d = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`grid`	STRING - Target grid description file or name
`map3d`	BOOL - Generate all mapfiles of the first 3D field
`ofile`	String with the path to the output file.

Details

```
remapnn  Nearest neighbor remapping
         Performs a nearest neighbor remapping on all input fields.
gennn    Generate nearest neighbor remap weights
      Generates nearest neighbor remapping weights for the first input field and writes the result to a f
         The format of this file is NetCDF following the SCRIP convention.
      Use the operator remap to apply this remapping weights to a data file with the same source grid.
      Set the parameter map3d=true to generate all mapfiles of the first 3D field with varying masks.
      In this case the mapfiles will be named &lt;outfile&gt;&lt;xxx&gt;.nc. xxx will have five digits w
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`remapstat` *Remaps source points to target cells*

Description

This module maps source points to target cells by calculating a statistical value from the source points. Each target cell contains the statistical value from all source points within that target cell. If there are no source points within a target cell, it gets a missing value. Depending on the chosen operator the minimum, maximum, range, sum, average, variance, standard deviation, skewness, kurtosis or median of source points is computed.

Usage

```
cdo_remapavg(ifile, grid = NULL, ofile = NULL)

cdo_remapkurt(ifile, grid = NULL, ofile = NULL)

cdo_remapmax(ifile, grid = NULL, ofile = NULL)

cdo_remapmean(ifile, grid = NULL, ofile = NULL)

cdo_remapmedian(ifile, grid = NULL, ofile = NULL)

cdo_remapmin(ifile, grid = NULL, ofile = NULL)

cdo_remaprange(ifile, grid = NULL, ofile = NULL)

cdo_remapskew(ifile, grid = NULL, ofile = NULL)

cdo_remapstd(ifile, grid = NULL, ofile = NULL)

cdo_remapstd1(ifile, grid = NULL, ofile = NULL)

cdo_remapsum(ifile, grid = NULL, ofile = NULL)

cdo_remapvar(ifile, grid = NULL, ofile = NULL)

cdo_remapvar1(ifile, grid = NULL, ofile = NULL)
```

Arguments

ifile	String with the path to the input file.
grid	STRING - Target grid description file or name
ofile	String with the path to the output file.

Details

```
remapmin     Remap minimum
             Minimum value of the source points.
remapmax     Remap maximum
             Maximum value of the source points.
remaprange   Remap range
             Range (max-min value) of the source points.
remapsum     Remap sum
             Sum of the source points.
remapmean    Remap mean
             Mean of the source points.
remapavg     Remap average
             Average of the source points.
remapstd     Remap standard deviation
             Standard deviation of the source points. Normalize by n.
remapstd1    Remap standard deviation (n-1)
             Standard deviation of the source points. Normalize by (n-1).
remapvar     Remap variance
             Variance of the source points. Normalize by n.
remapvar1    Remap variance (n-1)
             Variance of the source points. Normalize by (n-1).
remapskew    Remap skewness
             Skewness of the source points.
remapkurt    Remap kurtosis
             Kurtosis of the source points.
remapmedian  Remap median
             Median of the source points.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don’t return filenames return a character vector with the string output.

`replace` *Replace variables*

Description

This operator replaces variables in infile1 by variables from infile2 and write the result to outfile. Both input datasets need to have the same number of timesteps. All variable names may only occur once!

Usage

```
cdo_replace(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`replacevalues` *Replace variable values*

Description

This module replaces old variable values with new values, depending on the operator.

Usage

```
cdo_setrtoc(
  ifile,
  oldval = NULL,
  newval = NULL,
  rmin = NULL,
  rmax = NULL,
  c = NULL,
  c2 = NULL,
  ofile = NULL
)

cdo_setrtoc2(
  ifile,
  oldval = NULL,
  newval = NULL,
  rmin = NULL,
  rmax = NULL,
  c = NULL,
  c2 = NULL,
  ofile = NULL
)

cdo_setvals(
  ifile,
  oldval = NULL,
  newval = NULL,
  rmin = NULL,
  rmax = NULL,
  c = NULL,
  c2 = NULL,
  ofile = NULL
)
```

Arguments

`ifile`	String with the path to the input file.
`oldval`	FLOAT - Pairs of old and new values
`newval`	FLOAT - Pairs of old and new values
`rmin`	FLOAT - Lower bound
`rmax`	FLOAT - Upper bound
`c`	FLOAT - New value - inside range
`c2`	FLOAT - New value - outside range
`ofile`	String with the path to the output file.

Details

```
setvals   Set list of old values to new values
          Supply a list of n pairs of old and new values.
setrtoc   Set range to constant
                   / c      if i(t,x) GE rmin AND i(t,x) LE rmax
          o(t,x) =
                   \\ i(t,x) if i(t,x) LT rmin AND i(t,x) GT rmax
setrtoc2  Set range to constant others to constant2
                   / c      if i(t,x) GE rmin AND i(t,x) LE rmax
          o(t,x) =
                   \\ c2     if i(t,x) LT rmin AND i(t,x) GT rmax
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don’t return filenames return a character vector with the string output.

`rhopot` *Calculates potential density*

Description

This is a special operator for the post processing of the ocean and sea ice model MPIOM. It calculates the sea water potential density (name=rhopoto; code=18). Required input fields are sea water in-situ temperature (name=to; code=20) and sea water salinity (name=sao; code=5). Pressure is calculated from the level information or can be specified by the optional parameter.

Usage

```
cdo_rhopot(ifile, pressure = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`pressure`	FLOAT - Pressure in bar (constant value assigned to all levels)
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`rotuvb` *Rotation*

Description

This is a special operator for datsets with wind components on a rotated grid, e.g. data from the regional model REMO. It performs a backward transformation of velocity components U and V from a rotated spherical system to a geographical system.

Usage

```
cdo_rotuvb(ifile, u = NULL, v = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`u`	STRING - Pairs of zonal and meridional velocity components (use variable names or code numbers)
`v`	STRING - Pairs of zonal and meridional velocity components (use variable names or code numbers)
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

Note

This is a specific implementation for data from the REMO model, it may not work with data from other sources.

`runpctl` *Running percentile values*

Description

This module computes running percentiles over a selected number of timesteps in infile. The time of outfile is determined by the time in the middle of all contributing timesteps of infile. This can be change with the CDO option –timestat_date <first|middle|last>. o(t+(nts-1)/2,x) = pth percentile {i(t,x), i(t+1,x), ..., i(t+nts-1,x)}

Usage

```
cdo_runpctl(ifile, p = NULL, nts = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`p`	FLOAT - Percentile number in {0, ..., 100}
`nts`	INTEGER - Number of timesteps
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`runstat` *Running statistics*

Description

This module computes running statistical values over a selected number of timesteps. Depending on the chosen operator the minimum, maximum, range, sum, average, variance or standard deviation of a selected number of consecutive timesteps read from infile is written to outfile. The time of outfile is determined by the time in the middle of all contributing timesteps of infile. This can be change with the CDO option –timestat_date <first|middle|last>.

Usage

```
cdo_runavg(ifile, nts = NULL, ofile = NULL)

cdo_runmax(ifile, nts = NULL, ofile = NULL)

cdo_runmean(ifile, nts = NULL, ofile = NULL)

cdo_runmin(ifile, nts = NULL, ofile = NULL)

cdo_runrange(ifile, nts = NULL, ofile = NULL)

cdo_runstd(ifile, nts = NULL, ofile = NULL)

cdo_runstd1(ifile, nts = NULL, ofile = NULL)

cdo_runsum(ifile, nts = NULL, ofile = NULL)

cdo_runvar(ifile, nts = NULL, ofile = NULL)

cdo_runvar1(ifile, nts = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`nts`	INTEGER - Number of timesteps
`ofile`	String with the path to the output file.

Details

```
runmin    Running minimum
          o(t+(nts-1)/2,x) = min\{i(t,x), i(t+1,x), ..., i(t+nts-1,x)\}
runmax    Running maximum
          o(t+(nts-1)/2,x) = max\{i(t,x), i(t+1,x), ..., i(t+nts-1,x)\}
runrange  Running range
          o(t+(nts-1)/2,x) = range\{i(t,x), i(t+1,x), ..., i(t+nts-1,x)\}
```

```
runsum    Running sum
          o(t+(nts-1)/2,x) = sum\{i(t,x), i(t+1,x), ..., i(t+nts-1,x)\}
runmean   Running mean
          o(t+(nts-1)/2,x) = mean\{i(t,x), i(t+1,x), ..., i(t+nts-1,x)\}
runavg    Running average
          o(t+(nts-1)/2,x) = avg\{i(t,x), i(t+1,x), ..., i(t+nts-1,x)\}
runstd    Running standard deviation
          Normalize by n.

          o(t+(nts-1)/2,x) = std\{i(t,x), i(t+1,x), ..., i(t+nts-1,x)\}
runstd1   Running standard deviation (n-1)
          Normalize by (n-1).

          o(t+(nts-1)/2,x) = std1\{i(t,x), i(t+1,x), ..., i(t+nts-1,x)\}
runvar    Running variance
          Normalize by n.

          o(t+(nts-1)/2,x) = var\{i(t,x), i(t+1,x), ..., i(t+nts-1,x)\}
runvar1   Running variance (n-1)
          Normalize by (n-1).

          o(t+(nts-1)/2,x) = var1\{i(t,x), i(t+1,x), ..., i(t+nts-1,x)\}
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`samplegrid` *Resample grid*

Description

This is a special operator for resampling the horizontal grid. No interpolation takes place. Resample factor=2 means every second grid point is removed. Only rectilinear and curvilinear source grids are supported by this operator.

Usage

```
cdo_samplegrid(ifile, factor = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`factor`	INTEGER - Resample factor, typically 2, which will half the resolution
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`seaspctl` *Seasonal percentile values*

Description

This operator computes percentiles over all timesteps in infile1 of the same season. The algorithm uses histograms with minimum and maximum bounds given in infile2 and infile3, respectively. The default number of histogram bins is 101. The default can be overridden by defining the environment variable CDO_PCTL_NBINS. The files infile2 and infile3 should be the result of corresponding seasmin and seasmax operations, respectively. The time of outfile is determined by the time in the middle of all contributing timesteps of infile1. This can be change with the CDO option –timestat_date <first|middle|last>. Be careful about the first and the last output timestep, they may be incorrect values if the seasons have incomplete timesteps. For every adjacent sequence t_1, ...,t_n of timesteps of the same season it is: o(t,x) = pth percentile {i(t',x), t1 < t' <= tn}

Usage

```
cdo_seaspctl(ifile1, ifile2, ifile3, p = NULL, ofile = NULL)
```

Arguments

`ifile1, ifile2, ifile3`	Strings with the path to the input files.
`p`	FLOAT - Percentile number in {0, ..., 100}
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`seasstat` *Seasonal statistics*

Description

This module computes statistical values over timesteps of the same meteorological season. Depending on the chosen operator the minimum, maximum, range, sum, average, variance or standard deviation of timesteps of the same season is written to outfile. The time of outfile is determined by the time in the middle of all contributing timesteps of infile. This can be change with the CDO option –timestat_date <first|middle|last>. Be careful about the first and the last output timestep, they may be incorrect values if the seasons have incomplete timesteps.

Usage

```
cdo_seasavg(ifile, ofile = NULL)

cdo_seasmax(ifile, ofile = NULL)

cdo_seasmean(ifile, ofile = NULL)

cdo_seasmin(ifile, ofile = NULL)

cdo_seasrange(ifile, ofile = NULL)

cdo_seasstd(ifile, ofile = NULL)

cdo_seasstd1(ifile, ofile = NULL)

cdo_seassum(ifile, ofile = NULL)

cdo_seasvar(ifile, ofile = NULL)

cdo_seasvar1(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Details

```
seasmin    Seasonal minimum
       For every adjacent sequence t_1, ...,t_n of timesteps of the same season it is:

           o(t,x) = min\{i(t',x), t1 &lt; t' &lt;= tn\}
seasmax    Seasonal maximum
       For every adjacent sequence t_1, ...,t_n of timesteps of the same season it is:
           o(t,x) = max\{i(t',x), t1 &lt; t' &lt;= tn\}
seasrange   Seasonal range
       For every adjacent sequence t_1, ...,t_n of timesteps of the same season it is:

           o(t,x) = range\{i(t',x), t1 &lt; t' &lt;= tn\}
seassum     Seasonal sum
       For every adjacent sequence t_1, ...,t_n of timesteps of the same season it is:

           o(t,x) = sum\{i(t',x), t1 &lt; t' &lt;= tn\}
seasmean    Seasonal mean
       For every adjacent sequence t_1, ...,t_n of timesteps of the same season it is:

           o(t,x) = mean\{i(t',x), t1 &lt; t' &lt;= tn\}
seasavg     Seasonal average
       For every adjacent sequence t_1, ...,t_n of timesteps of the same season it is:

           o(t,x) = avg\{i(t',x), t1 &lt; t' &lt;= tn\}
seasstd     Seasonal standard deviation
       Normalize by n. For every adjacent sequence t_1, ...,t_n of timesteps of the same season it is:

           o(t,x) = std\{i(t',x), t1 &lt; t' &lt;= tn\}
seasstd1    Seasonal standard deviation (n-1)
       Normalize by (n-1). For every adjacent sequence t_1, ...,t_n of timesteps of the same season it i

           o(t,x) = std1\{i(t',x), t1 &lt; t' &lt;= tn\}
seasvar     Seasonal variance
       Normalize by n. For every adjacent sequence t_1, ...,t_n of timesteps of the same season it is:

           o(t,x) = var\{i(t',x), t1 &lt; t' &lt;= tn\}
seasvar1    Seasonal variance (n-1)
       Normalize by (n-1). For every adjacent sequence t_1, ...,t_n of timesteps of the same season it i

           o(t,x) = var1\{i(t',x), t1 &lt; t' &lt;= tn\}
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

---

selbox *Select a box*

Description

Selects grid cells inside a lon/lat or index box.

Usage

```
cdo_selindexbox(
  ifile,
  lon1 = NULL,
  lon2 = NULL,
  lat1 = NULL,
  lat2 = NULL,
  idx1 = NULL,
  idx2 = NULL,
  idy1 = NULL,
  idy2 = NULL,
  ofile = NULL
)

cdo_sellonlatbox(
  ifile,
  lon1 = NULL,
  lon2 = NULL,
  lat1 = NULL,
  lat2 = NULL,
  idx1 = NULL,
  idx2 = NULL,
  idy1 = NULL,
  idy2 = NULL,
  ofile = NULL
)
```

Arguments

`ifile`	String with the path to the input file.
`lon1`	FLOAT - Western longitude in degrees
`lon2`	FLOAT - Eastern longitude in degrees
`lat1`	FLOAT - Southern or northern latitude in degrees
`lat2`	FLOAT - Northern or southern latitude in degrees
`idx1`	INTEGER - Index of first longitude (1 - nlon)
`idx2`	INTEGER - Index of last longitude (1 - nlon)
`idy1`	INTEGER - Index of first latitude (1 - nlat)
`idy2`	INTEGER - Index of last latitude (1 - nlat)
`ofile`	String with the path to the output file.

Details

```
sellonlatbox  Select a longitude/latitude box
         Selects grid cells inside a lon/lat box. The user must specify the longitude and latitude of th
         Only those grid cells are considered whose grid center lies within the lon/lat box.
         For rotated lon/lat grids the parameters must be specified in rotated coordinates.
```

selindexbox   Select an index box
    Selects grid cells within an index box. The user must specify the indices of the edges of the bo
    The index of the left edge can be greater then the one of the right edge. Use negative indexing
    start from the end. The input grid must be a regular lon/lat or a 2D curvilinear grid.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`select` *Select fields*

Description

This module selects some fields from infiles and writes them to outfile. infiles is an arbitrary number of input files. All input files need to have the same structure with the same variables on different timesteps. The fields selected depends on the chosen parameters. Parameter is a comma-separated list of "key=value" pairs. A range of integer values can be specified by first/last[/inc]. Wildcards are supported for string values.

Usage

```
cdo_delete(
  ifiles,
  name = NULL,
  param = NULL,
  code = NULL,
  level = NULL,
  levrange = NULL,
  levidx = NULL,
  zaxisname = NULL,
  zaxisnum = NULL,
  ltype = NULL,
  gridname = NULL,
  gridnum = NULL,
  steptype = NULL,
  date = NULL,
  startdate = NULL,
  enddate = NULL,
  minute = NULL,
  hour = NULL,
  day = NULL,
  month = NULL,
  season = NULL,
  year = NULL,
  dom = NULL,
  timestep = NULL,
  timestep_of_year = NULL,
  timestepmask = NULL,
  ofile = NULL
)

cdo_select(
  ifiles,
  name = NULL,
  param = NULL,
  code = NULL,
  level = NULL,
  levrange = NULL,
  levidx = NULL,
  zaxisname = NULL,
  zaxisnum = NULL,
  ltype = NULL,
  gridname = NULL,
  gridnum = NULL,
  steptype = NULL,
  date = NULL,
  startdate = NULL,
  enddate = NULL,
  minute = NULL,
  hour = NULL,
  day = NULL,
  month = NULL,
  season = NULL,
  year = NULL,
  dom = NULL,
  timestep = NULL,
  timestep_of_year = NULL,
  timestepmask = NULL,
  ofile = NULL
)
```

Arguments

`ifiles`	Character vector with the path to the input files.
`name`	STRING - Comma-separated list of variable names.
`param`	STRING - Comma-separated list of parameter identifiers.
`code`	INTEGER - Comma-separated list or first/last[/inc] range of code numbers.
`level`	FLOAT - Comma-separated list of vertical levels.
`levrange`	FLOAT - First and last value of the level range.
`levidx`	INTEGER - Comma-separated list or first/last[/inc] range of index of levels.

`zaxisname`	STRING - Comma-separated list of zaxis names.
`zaxisnum`	INTEGER - Comma-separated list or first/last[/inc] range of zaxis numbers.
`ltype`	INTEGER - Comma-separated list or first/last[/inc] range of GRIB level types.
`gridname`	STRING - Comma-separated list of grid names.
`gridnum`	INTEGER - Comma-separated list or first/last[/inc] range of grid numbers.
`steptype`	STRING - Comma-separated list of timestep types (constant|avg|accum|min|max|range|diff|sum)
`date`	STRING - Comma-separated list of dates (format: YYYY-MM-DDThh:mm:ss).
`startdate`	STRING - Start date (format: YYYY-MM-DDThh:mm:ss).
`enddate`	STRING - End date (format: YYYY-MM-DDThh:mm:ss).
`minute`	INTEGER - Comma-separated list or first/last[/inc] range of minutes.
`hour`	INTEGER - Comma-separated list or first/last[/inc] range of hours.
`day`	INTEGER - Comma-separated list or first/last[/inc] range of days.
`month`	INTEGER - Comma-separated list or first/last[/inc] range of months.
`season`	STRING - Comma-separated list of seasons (substring of DJFMAMJJASOND or ANN).
`year`	INTEGER - Comma-separated list or first/last[/inc] range of years.
`dom`	STRING - Comma-separated list of the day of month (e.g. 29feb).
`timestep`	INTEGER - Comma-separated list or first/last[/inc] range of timesteps. Negative values select timesteps from the end (NetCDF only).
`timestep_of_year`	INTEGER - Comma-separated list or first/last[/inc] range of timesteps of year.
`timestepmask`	STRING - Read timesteps from a mask file.
`ofile`	String with the path to the output file.

Details

```
select  Select fields
        Selects all fields with parameters in a user given list.
delete  Delete fields
        Deletes all fields with parameters in a user given list.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`selgridcell` *Select grid cells*

Description

The operator selects grid cells of all fields from infile. The user must specify the index of each grid cell. The resulting grid in outfile is unstructured.

Usage

```
cdo_delgridcell(ifile, indices = NULL, ofile = NULL)

cdo_selgridcell(ifile, indices = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`indices`	INTEGER - Comma-separated list or first/last[/inc] range of indices
`ofile`	String with the path to the output file.

Details

```
selgridcell  Select grid cells
delgridcell  Delete grid cells
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`selmulti` *Select multiple fields via GRIB1 parameters*

Description

This module selects multiple fields from infile and writes them to outfile. selection-specification is a filename or in-place string with the selection specification. Each selection-specification has the following compact notation format: <type>(parameters; leveltype(s); levels) type " " sel for select or del for delete (optional) parameters" " GRIB1 parameter code number leveltype " " GRIB1 level type levels " " value of each level Examples: (1; 103; 0) (33,34; 105; 10) (11,17; 105; 2) (71,73,74,75,61,62,65,117,67,122,121,11,131,66,84,111,112; 105; 0) The following descriptive notation can also be used for selection specification from a file: SELECT/DELETE, PARAMETER=parameters, LEVTYPE=leveltye(s), LEVEL=levels Examples: SELECT, PARAMETER=1, LEVTYPE=103, LEVEL=0 SELECT, PARAMETER=33/34, LEVTYPE=105, LEVEL=10 SELECT, PARAMETER=11/17, LEVTYPE=105, LEVEL=2 SELECT, PARAMETER=71/73/74/75/61/62/65/117/67/122, LEVTYPE=105, LEVEL=0 DELETE, PARAMETER=128, LEVTYPE=109, LEVEL=* The following will convert Pressure from Pa into hPa; Temp from Kelvin to Celsius: SELECT, PARAMETER=1, LEVTYPE= 103, LEVEL=0, SCALE=0.01 SELECT, PARAMETER=11, LEVTYPE=105, LEVEL=2, OFFSET=273.15 If SCALE and/or OFFSET are defined, then the data values are scaled as SCALE*(VALUE-OFFSET).

Usage

```
cdo_changemulti(ifile, ofile = NULL)

cdo_delmulti(ifile, ofile = NULL)

cdo_selmulti(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Details

```
selmulti     Select multiple fields
delmulti     Delete multiple fields
changemulti  Change identication of multiple fields
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`selregion` *Select horizontal regions*

Description

Selects all grid cells with the center point inside user defined regions or a circle. The resulting grid is unstructured.

Usage

```
cdo_selcircle(
  ifile,
  regions = NULL,
  lon = NULL,
  lat = NULL,
  radius = NULL,
  ofile = NULL
)

cdo_selregion(
  ifile,
  regions = NULL,
  lon = NULL,
  lat = NULL,
  radius = NULL,
  ofile = NULL
)
```

Arguments

`ifile`	String with the path to the input file.
`regions`	STRING - Comma-separated list of ASCII formatted files with different regions
`lon`	FLOAT - Longitude of the center of the circle in degrees, default lon=0.0
`lat`	FLOAT - Latitude of the center of the circle in degrees, default lat=0.0
`radius`	STRING - Radius of the circle, default radius=1deg (units: deg, rad, km, m)
`ofile`	String with the path to the output file.

Details

```
selregion  Select cells inside regions
           Selects all grid cells with the center point inside the regions.
           Regions can be defined by the user via an ASCII file.
           Each region consists of the geographic coordinates of a polygon.
       Each line of a polygon description file contains the longitude and latitude of one point.
       Each polygon description file can contain one or more polygons separated by a line with the chara

         Predefined regions of countries can be specified via the country codes.
       A country is specified with dcw:&lt;CountryCode&gt;. Country codes can be combined with the plus
selcircle  Select cells inside a circle
       Selects all grid cells with the center point inside a circle. The circle is described by geograph
           of the center and the radius of the circle.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`selsurface` *Extract surface*

Description

This module computes a surface from all 3D variables. The result is a horizonal 2D field.

Usage

```
cdo_bottomvalue(ifile, isovalue = NULL, ofile = NULL)

cdo_isosurface(ifile, isovalue = NULL, ofile = NULL)

cdo_topvalue(ifile, isovalue = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`isovalue`	FLOAT - Isosurface value
`ofile`	String with the path to the output file.

Details

```
bottomvalue  Extract bottom level
                  This operator selects the valid values at the bottom level.
         The NetCDF CF compliant attribute positive is used to determine where top and bottom are.
         If this attribute is missing, low values are bottom and high values are top.
topvalue     Extract top level
                  This operator selects the valid values at the top level.
         The NetCDF CF compliant attribute positive is used to determine where top and bottom are.
         If this attribute is missing, low values are bottom and high values are top.
isosurface   Extract isosurface
         This operator computes an isosurface. The value of the isosurfce is specified by the parameter i
          The isosurface is calculated by linear interpolation between two layers.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

seltime *Select timesteps*

Description

This module selects user specified timesteps from infile and writes them to outfile. The timesteps selected depends on the chosen operator and the parameters. A range of integer values can be specified by first/last[/inc].

Usage

```
cdo_seldate(
  ifile,
  timesteps = NULL,
  times = NULL,
  hours = NULL,
  days = NULL,
  months = NULL,
  years = NULL,
  seasons = NULL,
  startdate = NULL,
  enddate = NULL,
  nts1 = NULL,
  nts2 = NULL,
  ofile = NULL
)

cdo_selday(
  ifile,
  timesteps = NULL,
  times = NULL,
  hours = NULL,
  days = NULL,
  months = NULL,
  years = NULL,
  seasons = NULL,
  startdate = NULL,
  enddate = NULL,
  nts1 = NULL,
  nts2 = NULL,
  ofile = NULL
)

cdo_selhour(
  ifile,
  timesteps = NULL,
  times = NULL,
  hours = NULL,
  days = NULL,
  months = NULL,
  years = NULL,
  seasons = NULL,
  startdate = NULL,
  enddate = NULL,
  nts1 = NULL,
  nts2 = NULL,
  ofile = NULL
)

cdo_selmonth(
  ifile,
  timesteps = NULL,
  times = NULL,
  hours = NULL,
  days = NULL,
  months = NULL,
  years = NULL,
  seasons = NULL,
  startdate = NULL,
  enddate = NULL,
  nts1 = NULL,
  nts2 = NULL,
  ofile = NULL
)

cdo_selseason(
  ifile,
  timesteps = NULL,
  times = NULL,
  hours = NULL,
  days = NULL,
  months = NULL,
  years = NULL,
  seasons = NULL,
  startdate = NULL,
  enddate = NULL,
  nts1 = NULL,
  nts2 = NULL,
  ofile = NULL
)

cdo_selsmon(
  ifile,
  timesteps = NULL,
  times = NULL,
  hours = NULL,
  days = NULL,
  months = NULL,
  years = NULL,
  seasons = NULL,
  startdate = NULL,
  enddate = NULL,
  nts1 = NULL,
  nts2 = NULL,
  ofile = NULL
)

cdo_seltime(
  ifile,
  timesteps = NULL,
  times = NULL,
  hours = NULL,
  days = NULL,
  months = NULL,
  years = NULL,
  seasons = NULL,
  startdate = NULL,
  enddate = NULL,
  nts1 = NULL,
  nts2 = NULL,
  ofile = NULL
)

cdo_seltimestep(
  ifile,
  timesteps = NULL,
  times = NULL,
  hours = NULL,
  days = NULL,
  months = NULL,
  years = NULL,
  seasons = NULL,
  startdate = NULL,
  enddate = NULL,
  nts1 = NULL,
  nts2 = NULL,
  ofile = NULL
)

cdo_selyear(
  ifile,
  timesteps = NULL,
  times = NULL,
  hours = NULL,
  days = NULL,
  months = NULL,
  years = NULL,
  seasons = NULL,
  startdate = NULL,
  enddate = NULL,
  nts1 = NULL,
  nts2 = NULL,
  ofile = NULL
)
```

Arguments

`ifile`	String with the path to the input file.
`timesteps`	INTEGER - Comma-separated list or first/last[/inc] range of timesteps. Negative values select timesteps from the end (NetCDF only).
`times`	STRING - Comma-separated list of times (format hh:mm:ss).
`hours`	INTEGER - Comma-separated list or first/last[/inc] range of hours.
`days`	INTEGER - Comma-separated list or first/last[/inc] range of days.
`months`	INTEGER - Comma-separated list or first/last[/inc] range of months.
`years`	INTEGER - Comma-separated list or first/last[/inc] range of years.
`seasons`	STRING - Comma-separated list of seasons (substring of DJFMAMJJASOND or ANN).
`startdate`	STRING - Start date (format: YYYY-MM-DDThh:mm:ss).
`enddate`	STRING - End date (format: YYYY-MM-DDThh:mm:ss) [default: startdate].
`nts1`	INTEGER - Number of timesteps before the selected month [default: 0].
`nts2`	INTEGER - Number of timesteps after the selected month [default: nts1].
`ofile`	String with the path to the output file.

Details

```
seltimestep  Select timesteps
            Selects all timesteps with a timestep in a user given list or range.
seltime      Select times
             Selects all timesteps with a time in a user given list or range.
selhour      Select hours
             Selects all timesteps with a hour in a user given list or range.
selday       Select days
             Selects all timesteps with a day in a user given list or range.
selmonth     Select months
             Selects all timesteps with a month in a user given list or range.
selyear      Select years
             Selects all timesteps with a year in a user given list or range.
selseason    Select seasons
```

```
            Selects all timesteps with a month of a season in a user given list.
seldate      Select dates
             Selects all timesteps with a date in a user given range.
selsmon      Select single month
        Selects a month and optional an arbitrary number of timesteps before and after this month.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`seltimeidx` *Select timestep by index*

Description

Selects field elements from infile2 according to a timestep index from infile1. The index of the timestep in infile1 should be the result of corresponding timminidx or timmaxidx operations, respectively.

Usage

```
cdo_seltimeidx(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

selvar *Select fields*

Description

This module selects some fields from infile and writes them to outfile. The fields selected depends on the chosen operator and the parameters. A range of integer values can be specified by first/last[/inc].

Usage

```
cdo_delcode(
  ifile,
  parameter = NULL,
  codes = NULL,
  names = NULL,
  stdnames = NULL,
  levels = NULL,
  levidx = NULL,
  ltypes = NULL,
  grids = NULL,
  zaxes = NULL,
  zaxisnames = NULL,
  tabnums = NULL,
  ofile = NULL
)

cdo_delname(
  ifile,
  parameter = NULL,
  codes = NULL,
  names = NULL,
  stdnames = NULL,
  levels = NULL,
  levidx = NULL,
  ltypes = NULL,
  grids = NULL,
  zaxes = NULL,
  zaxisnames = NULL,
  tabnums = NULL,
  ofile = NULL
)

cdo_delparam(
  ifile,
  parameter = NULL,
  codes = NULL,
  names = NULL,
  stdnames = NULL,
  levels = NULL,
  levidx = NULL,
  ltypes = NULL,
  grids = NULL,
  zaxes = NULL,
  zaxisnames = NULL,
  tabnums = NULL,
  ofile = NULL
)

cdo_selcode(
  ifile,
  parameter = NULL,
  codes = NULL,
  names = NULL,
  stdnames = NULL,
  levels = NULL,
  levidx = NULL,
  ltypes = NULL,
  grids = NULL,
  zaxes = NULL,
  zaxisnames = NULL,
  tabnums = NULL,
  ofile = NULL
)

cdo_selgrid(
  ifile,
  parameter = NULL,
  codes = NULL,
  names = NULL,
  stdnames = NULL,
  levels = NULL,
  levidx = NULL,
  ltypes = NULL,
  grids = NULL,
  zaxes = NULL,
  zaxisnames = NULL,
  tabnums = NULL,
  ofile = NULL
)

cdo_sellevel(
  ifile,
  parameter = NULL,
  codes = NULL,
  names = NULL,
  stdnames = NULL,
  levels = NULL,
  levidx = NULL,
  ltypes = NULL,
  grids = NULL,
  zaxes = NULL,
  zaxisnames = NULL,
  tabnums = NULL,
  ofile = NULL
)

cdo_sellevidx(
  ifile,
  parameter = NULL,
  codes = NULL,
  names = NULL,
  stdnames = NULL,
  levels = NULL,
  levidx = NULL,
  ltypes = NULL,
  grids = NULL,
  zaxes = NULL,
  zaxisnames = NULL,
  tabnums = NULL,
  ofile = NULL
)

cdo_selltype(
  ifile,
  parameter = NULL,
  codes = NULL,
  names = NULL,
  stdnames = NULL,
  levels = NULL,
  levidx = NULL,
  ltypes = NULL,
  grids = NULL,
  zaxes = NULL,
  zaxisnames = NULL,
  tabnums = NULL,
  ofile = NULL
)

cdo_selname(
  ifile,
  parameter = NULL,
  codes = NULL,
  names = NULL,
  stdnames = NULL,
  levels = NULL,
  levidx = NULL,
  ltypes = NULL,
  grids = NULL,
  zaxes = NULL,
  zaxisnames = NULL,
  tabnums = NULL,
  ofile = NULL
)

cdo_selparam(
  ifile,
  parameter = NULL,
  codes = NULL,
  names = NULL,
  stdnames = NULL,
  levels = NULL,
  levidx = NULL,
  ltypes = NULL,
  grids = NULL,
  zaxes = NULL,
  zaxisnames = NULL,
  tabnums = NULL,
  ofile = NULL
)

cdo_selstdname(
  ifile,
  parameter = NULL,
  codes = NULL,
  names = NULL,
  stdnames = NULL,
  levels = NULL,
  levidx = NULL,
  ltypes = NULL,
  grids = NULL,
  zaxes = NULL,
  zaxisnames = NULL,
  tabnums = NULL,
  ofile = NULL
)

cdo_seltabnum(
  ifile,
  parameter = NULL,
  codes = NULL,
  names = NULL,
  stdnames = NULL,
  levels = NULL,
  levidx = NULL,
  ltypes = NULL,
  grids = NULL,
  zaxes = NULL,
  zaxisnames = NULL,
  tabnums = NULL,
  ofile = NULL
)

cdo_selzaxis(
  ifile,
  parameter = NULL,
  codes = NULL,
  names = NULL,
  stdnames = NULL,
  levels = NULL,
  levidx = NULL,
  ltypes = NULL,
  grids = NULL,
  zaxes = NULL,
  zaxisnames = NULL,
  tabnums = NULL,
  ofile = NULL
)

cdo_selzaxisname(
  ifile,
  parameter = NULL,
  codes = NULL,
  names = NULL,
  stdnames = NULL,
  levels = NULL,
  levidx = NULL,
  ltypes = NULL,
  grids = NULL,
  zaxes = NULL,
  zaxisnames = NULL,
  tabnums = NULL,
  ofile = NULL
)
```

Arguments

`ifile`	String with the path to the input file.
`parameter`	STRING - Comma-separated list of parameter identifiers.

codes	INTEGER - Comma-separated list or first/last[/inc] range of code numbers.
names	STRING - Comma-separated list of variable names.
stdnames	STRING - Comma-separated list of standard names.
levels	FLOAT - Comma-separated list of vertical levels.
levidx	INTEGER - Comma-separated list or first/last[/inc] range of index of levels.
ltypes	INTEGER - Comma-separated list or first/last[/inc] range of GRIB level types.
grids	STRING - Comma-separated list of grid names or numbers.
zaxes	STRING - Comma-separated list of z-axis types or numbers.
zaxisnames	STRING - Comma-separated list of z-axis names.
tabnums	INTEGER - Comma-separated list or range of parameter table numbers.
ofile	String with the path to the output file.

Details

```
selparam      Select parameters by identifier
            Selects all fields with parameter identifiers in a user given list.
delparam      Delete parameters by identifier
            Deletes all fields with parameter identifiers in a user given list.
selcode       Select parameters by code number
            Selects all fields with code numbers in a user given list or range.
delcode       Delete parameters by code number
            Deletes all fields with code numbers in a user given list or range.
selname       Select parameters by name
              Selects all fields with parameter names in a user given list.
delname       Delete parameters by name
              Deletes all fields with parameter names in a user given list.
selstdname    Select parameters by standard name
              Selects all fields with standard names in a user given list.
sellevel      Select levels
              Selects all fields with levels in a user given list.
sellevidx     Select levels by index
           Selects all fields with index of levels in a user given list or range.
selgrid       Select grids
              Selects all fields with grids in a user given list.
selzaxis      Select z-axes
              Selects all fields with z-axes in a user given list.
selzaxisname  Select z-axes by name
              Selects all fields with z-axis names in a user given list.
selltype      Select GRIB level types
           Selects all fields with GRIB level type in a user given list or range.
seltabnum     Select parameter table numbers
         Selects all fields with parameter table numbers in a user given list or range.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

selyearidx *Select year by index*

Description

Selects field elements from infile2 according to a year index from infile1. The index of the year in infile1 should be the result of corresponding yearminidx or yearmaxidx operations, respectively.

Usage

```
cdo_selyearidx(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

set *Set field info*

Description

This module sets some field information. Depending on the chosen operator the parameter table, code number, parameter identifier, variable name or level is set.

Usage

```
cdo_setcode(
  ifile,
  table = NULL,
  code = NULL,
  param = NULL,
  name = NULL,
  level = NULL,
  ltype = NULL,
  maxsteps = NULL,
  ofile = NULL
)

cdo_setcodetab(
  ifile,
  table = NULL,
  code = NULL,
  param = NULL,
  name = NULL,
  level = NULL,
  ltype = NULL,
  maxsteps = NULL,
  ofile = NULL
)

cdo_setlevel(
  ifile,
  table = NULL,
  code = NULL,
  param = NULL,
  name = NULL,
  level = NULL,
  ltype = NULL,
  maxsteps = NULL,
  ofile = NULL
)

cdo_setltype(
  ifile,
  table = NULL,
  code = NULL,
  param = NULL,
  name = NULL,
  level = NULL,
  ltype = NULL,
  maxsteps = NULL,
  ofile = NULL
)

cdo_setmaxsteps(
  ifile,
  table = NULL,
  code = NULL,
  param = NULL,
  name = NULL,
  level = NULL,
  ltype = NULL,
  maxsteps = NULL,
  ofile = NULL
)

cdo_setname(
  ifile,
  table = NULL,
  code = NULL,
  param = NULL,
  name = NULL,
  level = NULL,
  ltype = NULL,
  maxsteps = NULL,
  ofile = NULL
)

cdo_setparam(
  ifile,
  table = NULL,
  code = NULL,
  param = NULL,
  name = NULL,
  level = NULL,
  ltype = NULL,
  maxsteps = NULL,
  ofile = NULL
)

cdo_setunit(
  ifile,
  table = NULL,
  code = NULL,
  param = NULL,
  name = NULL,
  level = NULL,
  ltype = NULL,
  maxsteps = NULL,
  ofile = NULL
)
```

Arguments

`ifile`	String with the path to the input file.
`table`	STRING - Parameter table file or name
`code`	INTEGER - Code number
`param`	STRING - Parameter identifier (GRIB1: code[.tabnum]; GRIB2: num[.cat[.dis]])
`name`	STRING - Variable name
`level`	FLOAT - New level

`ltype`	INTEGER - GRIB level type
`maxsteps`	INTEGER - Maximum number of timesteps
`ofile`	String with the path to the output file.

Details

```
setcodetab   Set parameter code table
             Sets the parameter code table for all variables.
setcode      Set code number
             Sets the code number for all variables to the same given value.
setparam     Set parameter identifier
             Sets the parameter identifier of the first variable.
setname      Set variable name
             Sets the name of the first variable.
setunit      Set variable unit
             Sets the unit of the first variable.
setlevel     Set level
             Sets the first level of all variables.
setltype     Set GRIB level type
             Sets the GRIB level type of all variables.
setmaxsteps  Set max timesteps
             Sets maximum number of timesteps
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don’t return filenames return a character vector with the string output.

`setattribute` *Set attributes*

Description

This operator sets or deletes attributes of a dataset and writes the result to outfile. The new attributes are only available in outfile if the file format supports attributes. Each attribute has the following structure: [var_nm@]att_nm[:{s|d|i}]=[att_val|{[var_nm@]att_nm}] var_nm Variable name (optional). Example: pressure att_nm Attribute name. Example: units att_val Comma-separated list of attribute values. Example: pascal The value of var_nm is the name of the variable containing the attribute (named att_nm) that you want to set. Use wildcards to set the attribute att_nm to more than one variable. A value of var_nm of '*' will set the attribute att_nm to all data variables. If var_nm is missing then att_nm refers to a global attribute. The value of att_nm is the name of the attribute you want to set. For each attribute a string (att_nm:s), a double (att_nm:d) or an integer (att_nm:i) type can be defined. By default the native type is set. The value of att_val is the contents of the attribute att_nm. att_val may be a single value or one-dimensional array of elements. The type and the number of elements of an attribute will be detected automatically from the contents of the values. An already existing attribute att_nm will be overwritten or it will be removed if att_val is omitted. Alternatively, the values of an existing attribute can be copied. This attribute must then be enclosed in curly brackets. A special meaning has the attribute name FILE. If this is the 1st attribute then all attributes are read from a file specified in the value of att_val.

Usage

```
cdo_delattribute(ifile, attributes = NULL, ofile = NULL)

cdo_setattribute(ifile, attributes = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`attributes`	STRING - Comma-separated list of attributes.
`ofile`	String with the path to the output file.

Details

```
setattribute  Set attributes
delattribute  Delete attributes
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

Note

Attributes are evaluated by CDO when opening infile. Therefor the result of this operator is not available for other operators when this operator is used in chaining operators.

`setbox` *Set a box to constant*

Description

Sets a box of the rectangularly understood field to a constant value. The elements outside the box are untouched, the elements inside are set to the given constant. All input fields need to have the same horizontal grid.

Usage

```
cdo_setcindexbox(
  ifile,
  c = NULL,
  lon1 = NULL,
  lon2 = NULL,
  lat1 = NULL,
  lat2 = NULL,
  idx1 = NULL,
  idx2 = NULL,
  idy1 = NULL,
  idy2 = NULL,
  ofile = NULL
)

cdo_setclonlatbox(
  ifile,
  c = NULL,
  lon1 = NULL,
  lon2 = NULL,
  lat1 = NULL,
  lat2 = NULL,
  idx1 = NULL,
  idx2 = NULL,
  idy1 = NULL,
  idy2 = NULL,
  ofile = NULL
)
```

Arguments

`ifile`	String with the path to the input file.
`c`	FLOAT - Constant
`lon1`	FLOAT - Western longitude
`lon2`	FLOAT - Eastern longitude
`lat1`	FLOAT - Southern or northern latitude
`lat2`	FLOAT - Northern or southern latitude
`idx1`	INTEGER - Index of first longitude
`idx2`	INTEGER - Index of last longitude
`idy1`	INTEGER - Index of first latitude
`idy2`	INTEGER - Index of last latitude
`ofile`	String with the path to the output file.

Details

```
setclonlatbox  Set a longitude/latitude box to constant
             Sets the values of a longitude/latitude box to a constant value. The
            user has to give the longitudes and latitudes of the edges of the box.
setcindexbox   Set an index box to constant
             Sets the values of an index box to a constant value. The user has to
             give the indices of the edges of the box. The index of the left edge
               can be greater than the one of the right edge.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

---

setfilter *Set NetCDF4 filter*

Description

This operator sets the NetCDF4 filter specification for selected variables. Filters are mainly used to compress/decompress data. NetCDF4 uses the HDF5 plugins for filter support. To find the HDF5 plugins, the environment variable HDF5_PLUGIN_PATH must point to the directory with the installed plugins. The program may terminate unexpectedly if filters are used whose plug-ins are not found. A filter specification consists of the filterId and the filter parameters. CDO supports multiple filters connected with '|'. Here is a filter specification for bzip2 (filterId: 307) combined with szip (filterId:4): "307,9|4,32,32". Use the CDO option –filter instead of setfilter if all variables require the same filter. More information about NetCDF4 filters can be found in https://docs.unidata.ucar.edu/netcdf-c/current/filters.html.

Usage

```
cdo_setfilter(ifile, filename = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`filename`	STRING - Read filter specification per variable from file [format: varname=\"<filterspec>\"]
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`setgrid` *Set grid information*

Description

This module modifies the metadata of the horizontal grid. Depending on the chosen operator a new grid description is set, the coordinates are converted or the grid cell area is added.

Usage

```
cdo_setgrid(
  ifile,
  grid = NULL,
  gridtype = NULL,
  gridarea = NULL,
  gridmask = NULL,
  projparams = NULL,
  ofile = NULL
)

cdo_setgridarea(
  ifile,
  grid = NULL,
  gridtype = NULL,
  gridarea = NULL,
  gridmask = NULL,
  projparams = NULL,
  ofile = NULL
)

cdo_setgridmask(
  ifile,
  grid = NULL,
  gridtype = NULL,
  gridarea = NULL,
  gridmask = NULL,
  projparams = NULL,
  ofile = NULL
)

cdo_setgridtype(
  ifile,
  grid = NULL,
  gridtype = NULL,
  gridarea = NULL,
  gridmask = NULL,
  projparams = NULL,
  ofile = NULL
)

cdo_setprojparams(
  ifile,
  grid = NULL,
  gridtype = NULL,
  gridarea = NULL,
  gridmask = NULL,
  projparams = NULL,
  ofile = NULL
)
```

Arguments

`ifile`	String with the path to the input file.
`grid`	STRING - Grid description file or name
`gridtype`	STRING - Grid type (curvilinear, unstructured, regular, lonlat, projection or dereference)
`gridarea`	STRING - Data file, the first field is used as grid cell area
`gridmask`	STRING - Data file, the first field is used as grid mask
`projparams`	STRING - Proj library parameter (e.g.:+init=EPSG:3413)
`ofile`	String with the path to the output file.

Details

```
setgrid        Set grid
          Sets a new grid description. The input fields need to have the same grid size as the size
               of the target grid description.
setgridtype    Set grid type
          Sets the grid type of all input fields. The following grid types are available:
          curvilinear &quot;   &quot;   Converts a regular grid to a curvilinear grid
          unstructured&quot;   &quot;   Converts a regular or curvilinear grid to an unstructured grid
              dereference &quot;    &quot;    Dereference a reference to a grid
          regular   &quot;   &quot;   Linear interpolation of a reduced Gaussian grid to a regular Gaus
          regularnn  &quot;   &quot;   Nearest neighbor interpolation of a reduced Gaussian grid to a r
          lonlat    &quot;   &quot;   Converts a regular lonlat grid stored as a curvilinear grid back t
          projection  &quot;   &quot;   Removes the geographical coordinates if projection parameter av
setgridarea    Set grid cell area
          Sets the grid cell area. The parameter gridarea is the path to a data file,
          the first field is used as grid cell area. The input fields need to have the same
            grid size as the grid cell area. The grid cell area is used to compute
          the weights of each grid cell if needed by an operator, e.g. for fldmean.
setgridmask    Set grid mask
            Sets the grid mask. The parameter gridmask is the path to a data file,
          the first field is used as the grid mask. The input fields need to have the same
          grid size as the grid mask. The grid mask is used as the target grid mask for
```

```
                remapping, e.g. for remapbil.
setprojparams  Set proj params
        Sets the proj_params attribute of a projection. This attribute is used to compute
                geographic coordinates of a projecton with the proj library.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`setgridcell` *Set the value of a grid cell*

Description

This operator sets the value of the selected grid cells. The grid cells can be selected by a comma-separated list of grid cell indices or a mask. The mask is read from a data file, which may contain only one field. If no grid cells are selected, all values are set.

Usage

```
cdo_setgridcell(ifile, value = NULL, cell = NULL, mask = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`value`	FLOAT - Value of the grid cell
`cell`	INTEGER - Comma-separated list of grid cell indices
`mask`	STRING - Name of the data file which contains the mask
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`sethalo` *Set the bounds of a field*

Description

This operator sets the boundary in the east, west, south and north of the rectangular understood fields. Positive values of the parameters increase the boundary in the selected direction. Negative values decrease the field at the selected boundary. The new rows and columns are filled with the missing value. With the optional parameter value a different fill value can be used. Global cyclic fields are filled cyclically at the east and west borders, if the fill value is not set by the user. All input fields need to have the same horizontal grid.

Usage

```
cdo_sethalo(
  ifile,
  east = NULL,
  west = NULL,
  south = NULL,
  north = NULL,
  value = NULL,
  ofile = NULL
)
```

Arguments

`ifile`	String with the path to the input file.
`east`	INTEGER - East halo
`west`	INTEGER - West halo
`south`	INTEGER - South halo
`north`	INTEGER - North halo
`value`	FLOAT - Fill value (default is the missing value)
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

setmiss	*Set missing value*

Description

This module sets part of a field to missing value or missing values to a constant value. Which part of the field is set depends on the chosen operator.

Usage

```
cdo_setctomiss(
  ifile,
  neighbors = NULL,
  newmiss = NULL,
  c = NULL,
  rmin = NULL,
  rmax = NULL,
  ofile = NULL
)

cdo_setmisstoc(
  ifile,
  neighbors = NULL,
  newmiss = NULL,
  c = NULL,
  rmin = NULL,
  rmax = NULL,
  ofile = NULL
)

cdo_setmisstodis(
  ifile,
  neighbors = NULL,
  newmiss = NULL,
  c = NULL,
  rmin = NULL,
  rmax = NULL,
  ofile = NULL
)

cdo_setmisstonn(
  ifile,
  neighbors = NULL,
  newmiss = NULL,
  c = NULL,
  rmin = NULL,
  rmax = NULL,
  ofile = NULL
)

cdo_setmissval(
  ifile,
  neighbors = NULL,
  newmiss = NULL,
  c = NULL,
  rmin = NULL,
  rmax = NULL,
  ofile = NULL
)

cdo_setrtomiss(
  ifile,
  neighbors = NULL,
  newmiss = NULL,
  c = NULL,
  rmin = NULL,
  rmax = NULL,
  ofile = NULL
)

cdo_setvrange(
  ifile,
  neighbors = NULL,
  newmiss = NULL,
  c = NULL,
  rmin = NULL,
  rmax = NULL,
  ofile = NULL
)
```

Arguments

`ifile`	String with the path to the input file.
`neighbors`	INTEGER - Number of nearest neighbors
`newmiss`	FLOAT - New missing value
`c`	FLOAT - Constant
`rmin`	FLOAT - Lower bound
`rmax`	FLOAT - Upper bound
`ofile`	String with the path to the output file.

Details

```
setmissval    Set a new missing value
                       / newmiss   if i(t,x) EQ miss
```

```
              o(t,x) =
                       \\ i(t,x)    if i(t,x) NE miss
setctomiss    Set constant to missing value
                       / miss   if i(t,x) EQ c
              o(t,x) =
                       \\ i(t,x) if i(t,x) NE c
setmisstoc    Set missing value to constant
                       / c      if i(t,x) EQ miss
              o(t,x) =
                       \\ i(t,x) if i(t,x) NE miss
setrtomiss    Set range to missing value
                       / miss   if i(t,x) GE rmin AND i(t,x) LE rmax
              o(t,x) =
                       \\ i(t,x) if i(t,x) LT rmin OR  i(t,x) GT rmax
setvrange     Set valid range
                       / miss   if i(t,x) LT rmin OR  i(t,x) GT rmax
              o(t,x) =
                       \\ i(t,x) if i(t,x) GE rmin AND i(t,x) LE rmax
setmisstonn   Set missing value to nearest neighbor
              Set all missing values to the nearest non missing value.
                       / i(t,y) if i(t,x) EQ miss AND i(t,y) NE miss
              o(t,x) =
                       \\ i(t,x) if i(t,x) NE miss
setmisstodis  Set missing value to distance-weighted average
         Set all missing values to the distance-weighted average of the nearest non missing values.
              The default number of nearest neighbors is 4.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

---

setpartab *Set parameter table*

Description

This module transforms data and metadata of infile via a parameter table and writes the result to outfile. A parameter table is an ASCII formatted file with a set of parameter entries for each variable. Each new set have to start with "&parameter" and to end with "/". The following parameter table entries are supported: Entry & Type & Description name & WORD & Name of the variable out_name & WORD & New name of the variable param & WORD & Parameter identifier (GRIB1: code[.tabnum]; GRIB2: num[.cat[.dis]]) out_param & WORD & New parameter identifier type & WORD & Data type (real or double) standard_name & WORD & As defined in the CF standard name table long_name & STRING & Describing the variable units & STRING & Specifying the units for the variable comment & STRING & Information concerning the variable cell_methods & STRING & Information concerning calculation of means or climatologies cell_measures & STRING & Indicates the names of the variables containing cell areas and volumes filterspec & STRING & NetCDF4 filter specification missing_value & FLOAT & Specifying how missing data will be identified valid_min & FLOAT & Minimum valid value valid_max & FLOAT & Maximum valid value ok_min_mean_abs & FLOAT & Minimum absolute mean ok_max_mean_abs & FLOAT & Maximum absolute mean factor & FLOAT & Scale factor delete & INTEGER & Set to 1 to delete variable convert & INTEGER & Set to 1 to convert the unit if necessary Unsupported parameter table entries are stored as variable attributes. The search key for the variable depends on the operator. Use setpartabn to search variables by the name. This is typically used for NetCDF datasets. The operator setpartabp searches variables by the parameter ID.

Usage

```
cdo_setpartabn(ifile, table = NULL, convert = NULL, ofile = NULL)

cdo_setpartabp(ifile, table = NULL, convert = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`table`	STRING - Parameter table file or name
`convert`	STRING - Converts the units if necessary
`ofile`	String with the path to the output file.

Details

```
setpartabp  Set parameter table
            Search variables by the parameter identifier.
setpartabn  Set parameter table
            Search variables by name.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`settime` *Set time*

Description

This module sets the time axis or part of the time axis. Which part of the time axis is overwritten/created depends on the chosen operator. The number of time steps does not change.

Usage

```
cdo_setcalendar(
  ifile,
  day = NULL,
  month = NULL,
  year = NULL,
  units = NULL,
  date = NULL,
  time = NULL,
  inc = NULL,
  frequency = NULL,
  calendar = NULL,
  shiftValue = NULL,
  ofile = NULL
)

cdo_setdate(
  ifile,
  day = NULL,
  month = NULL,
  year = NULL,
  units = NULL,
  date = NULL,
  time = NULL,
  inc = NULL,
  frequency = NULL,
  calendar = NULL,
  shiftValue = NULL,
  ofile = NULL
)

cdo_setday(
  ifile,
  day = NULL,
  month = NULL,
  year = NULL,
  units = NULL,
  date = NULL,
  time = NULL,
  inc = NULL,
  frequency = NULL,
  calendar = NULL,
  shiftValue = NULL,
  ofile = NULL
)

cdo_setmon(
  ifile,
  day = NULL,
  month = NULL,
  year = NULL,
  units = NULL,
  date = NULL,
  time = NULL,
  inc = NULL,
  frequency = NULL,
  calendar = NULL,
  shiftValue = NULL,
  ofile = NULL
)

cdo_setreftime(
  ifile,
  day = NULL,
  month = NULL,
  year = NULL,
  units = NULL,
  date = NULL,
  time = NULL,
  inc = NULL,
  frequency = NULL,
  calendar = NULL,
  shiftValue = NULL,
  ofile = NULL
)

cdo_settaxis(
  ifile,
  day = NULL,
  month = NULL,
  year = NULL,
  units = NULL,
  date = NULL,
  time = NULL,
  inc = NULL,
  frequency = NULL,
  calendar = NULL,
  shiftValue = NULL,
  ofile = NULL
)

cdo_settbounds(
  ifile,
  day = NULL,
  month = NULL,
  year = NULL,
  units = NULL,
  date = NULL,
  time = NULL,
  inc = NULL,
  frequency = NULL,
  calendar = NULL,
  shiftValue = NULL,
  ofile = NULL
)

cdo_settime(
  ifile,
  day = NULL,
  month = NULL,
  year = NULL,
  units = NULL,
  date = NULL,
  time = NULL,
  inc = NULL,
  frequency = NULL,
  calendar = NULL,
  shiftValue = NULL,
  ofile = NULL
)

cdo_settunits(
  ifile,
  day = NULL,
  month = NULL,
  year = NULL,
  units = NULL,
  date = NULL,
  time = NULL,
  inc = NULL,
  frequency = NULL,
  calendar = NULL,
  shiftValue = NULL,
  ofile = NULL
)

cdo_setyear(
  ifile,
  day = NULL,
  month = NULL,
  year = NULL,
  units = NULL,
  date = NULL,
  time = NULL,
  inc = NULL,
  frequency = NULL,
  calendar = NULL,
  shiftValue = NULL,
  ofile = NULL
)

cdo_shifttime(
  ifile,
  day = NULL,
  month = NULL,
  year = NULL,
  units = NULL,
  date = NULL,
  time = NULL,
  inc = NULL,
  frequency = NULL,
  calendar = NULL,
  shiftValue = NULL,
  ofile = NULL
)
```

Arguments

`ifile`	String with the path to the input file.
`day`	INTEGER - Value of the new day
`month`	INTEGER - Value of the new month
`year`	INTEGER - Value of the new year
`units`	STRING - Base units of the time axis (seconds|minutes|hours|days|months|years)
`date`	STRING - Date (format: YYYY-MM-DD)
`time`	STRING - Time (format: hh:mm:ss)
`inc`	STRING - Optional increment (seconds|minutes|hours|days|months|years) [default: 1hour]
`frequency`	STRING - Frequency of the time series (hour|day|month|year)
`calendar`	STRING - Calendar (standard|proleptic_gregorian|360_day|365_day|366_day)
`shiftValue`	STRING - Shift value (e.g. -3hour)
`ofile`	String with the path to the output file.

Details

```
setdate      Set date
             Sets the date in every timestep to the same given value.
settime      Set time of the day
             Sets the time in every timestep to the same given value.
setday       Set day
             Sets the day in every timestep to the same given value.
```

setmon	Set month Sets the month in every timestep to the same given value.
setyear	Set year Sets the year in every timestep to the same given value.
settunits	Set time units Sets the base units of a relative time axis.
settaxis	Set time axis Sets the time axis.
settbounds	Set time bounds Sets the time bounds.
setreftime	Set reference time Sets the reference time of a relative time axis.
setcalendar	Set calendar Sets the calendar attribute of a relative time axis.
shifttime	Shift timesteps Shifts all timesteps by the parameter shiftValue.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

---

setzaxis *Set zaxis information*

Description

This module modifies the metadata of the vertical grid.

Usage

```
cdo_genlevelbounds(ifile, zaxis = NULL, zbot = NULL, ztop = NULL, ofile = NULL)

cdo_setzaxis(ifile, zaxis = NULL, zbot = NULL, ztop = NULL, ofile = NULL)
```

Arguments

ifile	String with the path to the input file.
zaxis	STRING - Z-axis description file or name of the target z-axis
zbot	FLOAT - Specifying the bottom of the vertical column. Must have the same units as z-axis.
ztop	FLOAT - Specifying the top of the vertical column. Must have the same units as z-axis.
ofile	String with the path to the output file.

Details

```
setzaxis        Set z-axis
          This operator sets the z-axis description of all variables with the same number of level as th
genlevelbounds  Generate level bounds
                Generates the layer bounds of the z-axis.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`shiftxy` *Shift field*

Description

This module contains operators to shift all fields in x or y direction. All fields need to have the same horizontal rectilinear or curvilinear grid.

Usage

```
cdo_shiftx(ifile, nshift = NULL, cyclic = NULL, coord = NULL, ofile = NULL)

cdo_shifty(ifile, nshift = NULL, cyclic = NULL, coord = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`nshift`	INTEGER - Number of grid cells to shift (default: 1)
`cyclic`	STRING - If set, cells are filled up cyclic (default: missing value)
`coord`	STRING - If set, coordinates are also shifted
`ofile`	String with the path to the output file.

Details

```
shiftx  Shift x
        Shifts all fields in x direction.
shifty  Shift y
        Shifts all fields in y direction.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`showattribute` *Show attributes*

Description

This operator prints the attributes of the data variables of a dataset. Each attribute has the following structure: [var_nm@][att_nm] var_nm Variable name (optional). Example: pressure att_nm Attribute name (optional). Example: units The value of var_nm is the name of the variable containing the attribute (named att_nm) that you want to print. Use wildcards to print the attribute att_nm of more than one variable. A value of var_nm of '*' will print the attribute att_nm of all data variables. If var_nm is missing then att_nm refers to a global attribute. The value of att_nm is the name of the attribute you want to print. Use wildcards to print more than one attribute. A value of att_nm of '*'* will print all attributes.

Usage

```
cdo_showattribute(ifile, attributes = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`attributes`	STRING - Comma-separated list of attributes.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`showinfo` *Show variables, levels or times*

Description

This module prints the format, variables, levels or times of the input dataset.

Usage

```
cdo_showcode(ifile)

cdo_showdate(ifile)

cdo_showfilter(ifile)

cdo_showformat(ifile)
```

```
cdo_showlevel(ifile)

cdo_showltype(ifile)

cdo_showmon(ifile)

cdo_showname(ifile)

cdo_showstdname(ifile)

cdo_showtime(ifile)

cdo_showtimestamp(ifile)

cdo_showyear(ifile)
```

Arguments

`ifile`	String with the path to the input file.

Details

```
showformat     Show file format
               Prints the file format of the input dataset.
showcode       Show code numbers
               Prints the code number of all variables.
showname       Show variable names
               Prints the name of all variables.
showstdname    Show standard names
               Prints the standard name of all variables.
showlevel      Show levels
               Prints all levels for each variable.
showltype      Show GRIB level types
               Prints the GRIB level type for all z-axes.
showyear       Show years
               Prints all years.
showmon        Show months
               Prints all months.
showdate       Show date information
               Prints date information of all timesteps (format YYYY-MM-DD).
showtime       Show time information
               Prints time information of all timesteps (format hh:mm:ss).
showtimestamp  Show timestamp
               Prints timestamp of all timesteps (format YYYY-MM-DDThh:mm:ss).
showfilter     Show filter specification
               Prints NetCDF4 filter specification of all variables.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`sinfo` *Short information*

Description

This module writes information about the structure of infiles to standard output. infiles is an arbitrary number of input files. All input files need to have the same structure with the same variables on different timesteps. The information displayed depends on the chosen operator.

Usage

```
cdo_sinfo(ifiles)

cdo_sinfon(ifiles)
```

Arguments

`ifiles`	Character vector with the path to the input files.

Details

```
sinfo   Short information listed by parameter identifier
     Prints short information of a dataset. The information is divided into 4 sections.
        Section 1 prints one line per parameter with the following information:
        - institute and source
        - time c=constant v=varying
        - type of statistical processing
        - number of levels and z-axis number
        - horizontal grid size and number
        - data type
        - parameter identifier
       Section 2 and 3 gives a short overview of all grid and vertical coordinates.
        And the last section contains short information of the time coordinate.
sinfon  Short information listed by parameter name
     The same as operator sinfo but using the name instead of the identifier to label the parameter.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`smooth` *Smooth grid points*

Description

Smooth all grid points of a horizontal grid. Options is a comma-separated list of "key=value" pairs with optional parameters.

Usage

```
cdo_smooth(
  ifile,
  nsmooth = NULL,
  radius = NULL,
  maxpoints = NULL,
  weighted = NULL,
  weight0 = NULL,
  weightR = NULL,
  ofile = NULL
)

cdo_smooth9(
  ifile,
  nsmooth = NULL,
  radius = NULL,
  maxpoints = NULL,
  weighted = NULL,
  weight0 = NULL,
  weightR = NULL,
  ofile = NULL
)
```

Arguments

`ifile`	String with the path to the input file.
`nsmooth`	INTEGER - Number of times to smooth, default nsmooth=1
`radius`	STRING - Search radius, default radius=1deg (units: deg, rad, km, m)
`maxpoints`	INTEGER - Maximum number of points, default maxpoints=<gridsize>
`weighted`	STRING - Weighting method, default weighted=linear
`weight0`	FLOAT - Weight at distance 0, default weight0=0.25
`weightR`	FLOAT - Weight at the search radius, default weightR=0.25
`ofile`	String with the path to the output file.

Details

```
smooth   Smooth grid points
      Performs a N point smoothing on all input fields. The number of points used depend
       on the search radius (radius) and the maximum number of points (maxpoints).
         Per default all points within the search radius of 1degree are used.
       The weights for the points depend on the weighting method and the distance.
      The implemented weighting method is linear with constant default weights of 0.25
         at distance 0 (weight0) and at the search radius (weightR).
smooth9  9 point smoothing
      Performs a 9 point smoothing on all fields with a quadrilateral curvilinear grid.
        The result at each grid point is a weighted average of the grid point plus
         the 8 surrounding points. The center point receives a weight of 1.0, the
       points at each side and above and below receive a weight of 0.5, and corner
         points receive a weight of 0.3.
        All 9 points are multiplied by their weights and summed, then divided by
        the total weight to obtain the smoothed value. Any missing data points are
        not included in the sum; points beyond the grid boundary are considered to
      be missing. Thus the final result may be the result of an averaging with less
         than 9 points.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

---

specconv *Spectral conversion*

Description

Changed the triangular truncation of all spectral fields. This operator performs downward conversion by cutting the resolution. Upward conversions are achieved by filling in zeros.

Usage

```
cdo_sp2sp(ifile, trunc = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`trunc`	INTEGER - New spectral resolution
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`spectral` *Spectral transformation*

Description

This module transforms fields on a global regular Gaussian grid to spectral coefficients and vice versa. The transformation is achieved by applying Fast Fourier Transformation (FFT) first and direct Legendre Transformation afterwards in gp2sp. In sp2gp the inverse Legendre Transformation and inverse FFT are used. Missing values are not supported. The relationship between the spectral resolution, governed by the truncation number T, and the grid resolution depends on the number of grid points at which the shortest wavelength field is represented. For a grid with 2N points between the poles (so 4N grid points in total around the globe) the relationship is: linear grid: the shortest wavelength is represented by 2 grid points $\rightarrow$ 4N $\simeq$ 2(TL + 1) quadratic grid: the shortest wavelength is represented by 3 grid points $\rightarrow$ 4N $\simeq$ 3(TQ + 1) cubic grid: the shortest wavelength is represented by 4 grid points $\rightarrow$ 4N $\simeq$ 4(TC + 1) The quadratic grid is used by ECHAM and ERA15. ERA40 is using a linear Gaussian grid reflected by the TL notation. The following table shows the calculation of the number of latitudes and the triangular truncation for the different grid types: Gridtype & Number of latitudes: nlat & Triangular truncation: ntr linear & NINT((ntr*2 + 1)/2) & (nlat*2 - 1) / 2 quadratic & NINT((ntr*3 + 1)/2) & (nlat*2 - 1) / 3 cubic & NINT((ntr*4 + 1)/2) & (nlat*2 - 1) / 4

Usage

```
cdo_gp2sp(ifile, type = NULL, trunc = NULL, ofile = NULL)

cdo_sp2gp(ifile, type = NULL, trunc = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`type`	STRING - Type of the grid: quadratic, linear, cubic (default: type=quadratic)
`trunc`	STRING - Triangular truncation
`ofile`	String with the path to the output file.

Details

```
sp2gp  Spectral to gridpoint
       Convert all spectral fields to a global regular Gaussian grid.
       The optional parameter trunc must be greater than the input truncation.
gp2sp  Gridpoint to spectral
       Convert all Gaussian gridpoint fields to spectral fields.
       The optional parameter trunc must be lower than the input truncation.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

Note

To speed up the calculations, the Legendre polynoms are kept in memory. This requires a relatively large amount of memory. This is for example 12GB for T1279 data.

`split` *Split a dataset*

Description

This module splits infile into pieces. The output files will be named <obase><xxx><suffix> where suffix is the filename extension derived from the file format. xxx and the contents of the output files depends on the chosen operator. params is a comma-separated list of processing parameters.

Usage

```
cdo_splitcode(ifile, swap = NULL, uuid = NULL, obase = NULL)

cdo_splitgrid(ifile, swap = NULL, uuid = NULL, obase = NULL)

cdo_splitlevel(ifile, swap = NULL, uuid = NULL, obase = NULL)

cdo_splitname(ifile, swap = NULL, uuid = NULL, obase = NULL)

cdo_splitparam(ifile, swap = NULL, uuid = NULL, obase = NULL)

cdo_splittabnum(ifile, swap = NULL, uuid = NULL, obase = NULL)

cdo_splitzaxis(ifile, swap = NULL, uuid = NULL, obase = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`swap`	STRING - Swap the position of obase and xxx in the output filename
`uuid`	STRING - Add a UUID as global attribute <attname> to each output file
`obase`	String with the basename of the output files.

Details

`splitcode`	`Split code numbers`
	`Splits a dataset into pieces, one for each different code number.`
	`xxx will have three digits with the code number.`

`splitparam`	`Split parameter identifiers`
	`Splits a dataset into pieces, one for each different parameter identifier.`
	`xxx will be a string with the parameter identifier.`

`splitname`	`Split variable names`
	`Splits a dataset into pieces, one for each variable name.`
	`xxx will be a string with the variable name.`

`splitlevel`	`Split levels`
	`Splits a dataset into pieces, one for each different level.`
	`xxx will have six digits with the level.`

`splitgrid`	`Split grids`
	`Splits a dataset into pieces, one for each different grid.`
	`xxx will have two digits with the grid number.`

`splitzaxis`	`Split z-axes`
	`Splits a dataset into pieces, one for each different z-axis.`
	`xxx will have two digits with the z-axis number.`

`splittabnum`	`Split parameter table numbers`
	`Splits a dataset into pieces, one for each GRIB1 parameter table number.`
	`xxx will have three digits with the GRIB1 parameter table number.`

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

Note

Operators of this module need to open all output files simultaneously. The maximum number of open files depends on the operating system!

`splitdate` *Splits a file into dates*

Description

This operator splits infile into pieces, one for each different date. The output files will be named <obase><YYYY-MM-DD><suffix> where YYYY-MM-DD is the date and suffix is the filename extension derived from the file format.

Usage

```
cdo_splitdate(ifile, obase = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`obase`	String with the basename of the output files.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`splitsel` *Split selected timesteps*

Description

This operator splits infile into pieces, one for each adjacent sequence t_1, ..., t_n of timesteps of the same selected time range. The output files will be named <obase><nnnnnn><suffix> where nnnnnn is the sequence number and suffix is the filename extension derived from the file format.

Usage

```
cdo_splitsel(ifile, nsets = NULL, noffset = NULL, nskip = NULL, obase = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`nsets`	INTEGER - Number of input timesteps for each output file
`noffset`	INTEGER - Number of input timesteps skipped before the first timestep range (optional)
`nskip`	INTEGER - Number of input timesteps skipped between timestep ranges (optional)
`obase`	String with the basename of the output files.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

splittime — *Split timesteps of a dataset*

Description

This module splits infile into timesteps pieces. The output files will be named <obase><xxx><suffix> where suffix is the filename extension derived from the file format. xxx and the contents of the output files depends on the chosen operator.

Usage

```
cdo_splitday(ifile, format = NULL, obase = NULL)

cdo_splithour(ifile, format = NULL, obase = NULL)

cdo_splitmon(ifile, format = NULL, obase = NULL)

cdo_splitseas(ifile, format = NULL, obase = NULL)

cdo_splityear(ifile, format = NULL, obase = NULL)

cdo_splityearmon(ifile, format = NULL, obase = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`format`	STRING - C-style format for strftime() (e.g. %B for the full month name)
`obase`	String with the basename of the output files.

Details

```
splithour     Split hours
              Splits a file into pieces, one for each different hour.
              xxx will have two digits with the hour.
splitday      Split days
              Splits a file into pieces, one for each different day.
              xxx will have two digits with the day.
splitseas     Split seasons
              Splits a file into pieces, one for each different season.
              xxx will have three characters with the season.
splityear     Split years
              Splits a file into pieces, one for each different year.
              xxx will have four digits with the year (YYYY).
splityearmon  Split in years and months
              Splits a file into pieces, one for each different year and month.
              xxx will have six digits with the year and month (YYYYMM).
splitmon      Split months
```

```
Splits a file into pieces, one for each different month.
xxx will have two digits with the month.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

Note

Operators of this module need to open all output files simultaneously. The maximum number of open files depends on the operating system!

---

strbre *Strong breeze days index per time period*

Description

Let infile be a time series of the daily maximum horizontal wind speed VX, then the number of days where VX is greater than or equal to 10.5 m/s is counted. A further output variable is the maximum number of consecutive days with maximum wind speed greater than or equal to 10.5 m/s. Note that VX is defined as the square root of the sum of squares of the zonal and meridional wind speeds and have to be given in units of m/s. The date information of a timestep in outfile is the date of the last contributing timestep in infile.

Usage

```
cdo_strbre(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

strgal *Strong gale days index per time period*

Description

Let infile be a time series of the daily maximum horizontal wind speed VX, then the number of days where VX is greater than or equal to 20.5 m/s is counted. A further output variable is the maximum number of consecutive days with maximum wind speed greater than or equal to 20.5 m/s. Note that VX is defined as the square root of the sum of square of the zonal and meridional wind speeds and have to be given in units of m/s. The date information of a timestep in outfile is the date of the last contributing timestep in infile.

Usage

```
cdo_strgal(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

strwin *Strong wind days index per time period*

Description

Let infile be a time series of the daily maximum horizontal wind speed VX, then the number of days where VX > v is counted. The horizontal wind speed v is an optional parameter with default v = 10.5 m/s. A further output variable is the maximum number of consecutive days with maximum wind speed greater than or equal to v. Note that both VX and v have to be given in units of m/s. Also note that the horizontal wind speed is defined as the square root of the sum of squares of the zonal and meridional wind speeds. The date information of a timestep in outfile is the date of the last contributing timestep in infile.

Usage

```
cdo_strwin(ifile, v = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`v`	FLOAT - Horizontal wind speed threshold (m/s, default v = 10.5 m/s)
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`tee` *Duplicate a data stream and write it to file*

Description

This operator copies the input dataset to outfile1 and outfile2. The first output stream in outfile1 can be further processesd with other cdo operators. The second output outfile2 is written to disk. It can be used to store intermediate results to a file.

Usage

```
cdo_tee(ifile, outfile2 = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`outfile2`	STRING - Destination filename for the copy of the input file
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

timcor *Correlation over time*

Description

The correlation coefficient is a quantity that gives the quality of a least squares fitting to the original data. This operator correlates each gridpoint of two fields over all timesteps. If there is only one input field, the p-value (probability value) is also written out. With S(x) = {t, i_1(t,x) != missval and i_2(t,x) != missval} it is o(1,x) = Cor{(i_1(t,x), i_2(t,x)), t_1 < t <= t_n} For every gridpoint x only those timesteps t belong to the sample, which have i_1(t,x) != missval and i_2(t,x) != missval.

Usage

```
cdo_timcor(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

timcovar *Covariance over time*

Description

This operator calculates the covariance of two fields at each gridpoint over all timesteps. With S(x) = {t, i_1(t,x) != missval and i_2(t,x) != missval} it is o(1,x) = Covar{(i_1(t,x), i_2(t,x)), t_1 < t <= t_n} For every gridpoint x only those timesteps t belong to the sample, which have i_1(t,x) != missval and i_2(t,x) != missval.

Usage

```
cdo_timcovar(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`timcumsum` *Cumulative sum over all timesteps*

Description

The timcumsum operator calculates the cumulative sum over all timesteps. Missing values are treated as numeric zero when summing. o(t,x) = sum{i(t',x), 0<t'<=t}

Usage

```
cdo_timcumsum(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`timfillmiss` *Temporal filling of missing values*

Description

This operator fills in temporally missing values. The method parameter can be used to select the filling method. The default method=nearest fills missing values with the nearest neighbor value. Other options are forward and backward to fill missing values by forward or backward propagation of values. Use the limit parameter to set the maximum number of consecutive missing values to fill and max_gaps to set the maximum number of gaps to fill.

Usage

```
cdo_timfillmiss(
  ifile,
  method = NULL,
  limit = NULL,
  max_gaps = NULL,
  ofile = NULL
)
```

Arguments

`ifile`	String with the path to the input file.
`method`	STRING - Fill method [nearest|linear|forward|backward] (default: nearest)
`limit`	INTEGER - The maximum number of consecutive missing values to fill (default: all)
`max_gaps`	INTEGER - The maximum number of gaps to fill (default: all)
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`timpctl` *Percentile values over all timesteps*

Description

This operator computes percentiles over all timesteps in infile1. The algorithm uses histograms with minimum and maximum bounds given in infile2 and infile3, respectively. The default number of histogram bins is 101. The default can be overridden by defining the environment variable CDO_PCTL_NBINS. The files infile2 and infile3 should be the result of corresponding timmin and timmax operations, respectively. The time of outfile is determined by the time in the middle of all contributing timesteps of infile1. This can be change with the CDO option –timestat_date <first|middle|last>. o(1,x) = pth percentile {i(t',x), t_1<t'<=t_n}

Usage

```
cdo_timpctl(ifile1, ifile2, ifile3, p = NULL, ofile = NULL)
```

Arguments

`ifile1, ifile2, ifile3`	Strings with the path to the input files.
`p`	FLOAT - Percentile number in {0, ..., 100}
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`timselpctl` *Time range percentile values*

Description

This operator computes percentile values over a selected number of timesteps in infile1. The algorithm uses histograms with minimum and maximum bounds given in infile2 and infile3, respectively. The default number of histogram bins is 101. The default can be overridden by setting the environment variable CDO_PCTL_NBINS to a different value. The files infile2 and infile3 should be the result of corresponding timselmin and timselmax operations, respectively. The time of outfile is determined by the time in the middle of all contributing timesteps of infile1. This can be change with the CDO option –timestat_date <first|middle|last>. For every adjacent sequence t1, ..., tn of timesteps of the same selected time range it is: o(t,x) = pth percentile {i(t',x), t1 < t' <= tn}

Usage

```
cdo_timselpctl(
  ifile1,
  ifile2,
  ifile3,
  p = NULL,
  nsets = NULL,
  noffset = NULL,
  nskip = NULL,
  ofile = NULL
)
```

Arguments

`ifile1, ifile2, ifile3`	Strings with the path to the input files.
`p`	FLOAT - Percentile number in {0, ..., 100}
`nsets`	INTEGER - Number of input timesteps for each output timestep
`noffset`	INTEGER - Number of input timesteps skipped before the first timestep range (optional)
`nskip`	INTEGER - Number of input timesteps skipped between timestep ranges (optional)
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`timselstat` *Time range statistics*

Description

This module computes statistical values for a selected number of timesteps. According to the chosen operator the minimum, maximum, range, sum, average, variance or standard deviation of the selected timesteps is written to outfile. The time of outfile is determined by the time in the middle of all contributing timesteps of infile. This can be change with the CDO option –timestat_date <first|middle|last>.

Usage

```
cdo_timselavg(ifile, nsets = NULL, noffset = NULL, nskip = NULL, ofile = NULL)

cdo_timselmax(ifile, nsets = NULL, noffset = NULL, nskip = NULL, ofile = NULL)

cdo_timselmean(ifile, nsets = NULL, noffset = NULL, nskip = NULL, ofile = NULL)

cdo_timselmin(ifile, nsets = NULL, noffset = NULL, nskip = NULL, ofile = NULL)

cdo_timselrange(
  ifile,
  nsets = NULL,
  noffset = NULL,
  nskip = NULL,
  ofile = NULL
)

cdo_timselstd(ifile, nsets = NULL, noffset = NULL, nskip = NULL, ofile = NULL)

cdo_timselstd1(ifile, nsets = NULL, noffset = NULL, nskip = NULL, ofile = NULL)

cdo_timselsum(ifile, nsets = NULL, noffset = NULL, nskip = NULL, ofile = NULL)

cdo_timselvar(ifile, nsets = NULL, noffset = NULL, nskip = NULL, ofile = NULL)

cdo_timselvar1(ifile, nsets = NULL, noffset = NULL, nskip = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`nsets`	INTEGER - Number of input timesteps for each output timestep
`noffset`	INTEGER - Number of input timesteps skipped before the first timestep range (optional)
`nskip`	INTEGER - Number of input timesteps skipped between timestep ranges (optional)
`ofile`	String with the path to the output file.

Details

```
timselmin    Time selection minimum
        For every adjacent sequence t1, ...., tn of timesteps of the same selected time range it is:

            o(t,x) = min\{i(t',x), t1 &lt; t' &lt;= tn\}
timselmax    Time selection maximum
        For every adjacent sequence t1, ...., tn of timesteps of the same selected time range it is:

            o(t,x) = max\{i(t',x), t1 &lt; t' &lt;= tn\}
timselrange  Time selection range
        For every adjacent sequence t1, ...., tn of timesteps of the same selected time range it is:

            o(t,x) = range\{i(t',x), t1 &lt; t' &lt;= tn\}
timselsum    Time selection sum
        For every adjacent sequence t1, ...., tn of timesteps of the same selected time range it is:

            o(t,x) = sum\{i(t',x), t1 &lt; t' &lt;= tn\}
timselmean   Time selection mean
        For every adjacent sequence t1, ...., tn of timesteps of the same selected time range it is:

            o(t,x) = mean\{i(t',x), t1 &lt; t' &lt;= tn\}
timselavg    Time selection average
        For every adjacent sequence t1, ...., tn of timesteps of the same selected time range it is:

            o(t,x) = avg\{i(t',x), t1 &lt; t' &lt;= tn\}
timselstd    Time selection standard deviation
        Normalize by n. For every adjacent sequence t1, ...., tn of timesteps of the same selected time

            o(t,x) = std\{i(t',x), t1 &lt; t' &lt;= tn\}
timselstd1   Time selection standard deviation (n-1)
        Normalize by (n-1). For every adjacent sequence t1, ...., tn of timesteps of the same selected t

            o(t,x) = std1\{i(t',x), t1 &lt; t' &lt;= tn\}
timselvar    Time selection variance
        Normalize by n. For every adjacent sequence t1, ...., tn of timesteps of the same selected time

            o(t,x) = var\{i(t',x), t1 &lt; t' &lt;= tn\}
```

```
timselvar1   Time selection variance (n-1)
        Normalize by (n-1). For every adjacent sequence t1, ...., tn of timesteps of the same selected t

            o(t,x) = var1\{i(t',x), t1 &lt; t' &lt;= tn\}
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`timsort` *Timsort*

Description

Sorts the elements in ascending order over all timesteps for every field position. After sorting it is: o(t_1,x) <= o(t_2,x) forall (t_1<t_2),x

Usage

```
cdo_timsort(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`timstat`	*Statistical values over all timesteps*

Description

This module computes statistical values over all timesteps in infile. Depending on the chosen operator the minimum, maximum, range, sum, average, variance or standard deviation of all timesteps read from infile is written to outfile. The time of outfile is determined by the time in the middle of all contributing timesteps of infile. This can be change with the CDO option –timestat_date <first|middle|last>.

Usage

```
cdo_timavg(ifile, ofile = NULL)

cdo_timmax(ifile, ofile = NULL)

cdo_timmaxidx(ifile, ofile = NULL)

cdo_timmean(ifile, ofile = NULL)

cdo_timmin(ifile, ofile = NULL)

cdo_timminidx(ifile, ofile = NULL)

cdo_timrange(ifile, ofile = NULL)

cdo_timstd(ifile, ofile = NULL)

cdo_timstd1(ifile, ofile = NULL)

cdo_timsum(ifile, ofile = NULL)

cdo_timvar(ifile, ofile = NULL)

cdo_timvar1(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Details

```
timmin     Time minimum
           o(1,x) = min\{i(t',x), t_1&lt;t'&lt;=t_n\}
timmax     Time maximum
```

```
           o(1,x) = max\{i(t',x), t_1&lt;t'&lt;=t_n\}
timminidx  Index of time minimum
           o(1,x) = minidx\{i(t',x), t_1&lt;t'&lt;=t_n\}
timmaxidx  Index of time maximum
           o(1,x) = maxidx\{i(t',x), t_1&lt;t'&lt;=t_n\}
timrange   Time range
           o(1,x) = range\{i(t',x), t_1&lt;t'&lt;=t_n\}
timsum     Time sum
           o(1,x) = sum\{i(t',x), t_1&lt;t'&lt;=t_n\}
timmean    Time mean
           o(1,x) = mean\{i(t',x), t_1&lt;t'&lt;=t_n\}
timavg     Time average
           o(1,x) = avg\{i(t',x), t_1&lt;t'&lt;=t_n\}
timstd     Time standard deviation
           Normalize by n.

           o(1,x) = std\{i(t',x), t_1&lt;t'&lt;=t_n\}
timstd1    Time standard deviation (n-1)
           Normalize by (n-1).

           o(1,x) = std1\{i(t',x), t_1&lt;t'&lt;=t_n\}
timvar     Time variance
           Normalize by n.

           o(1,x) = var\{i(t',x), t_1&lt;t'&lt;=t_n\}
timvar1    Time variance (n-1)
           Normalize by (n-1).

           o(1,x) = var1\{i(t',x), t_1&lt;t'&lt;=t_n\}
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`trend` *Trend of time series*

Description

The values of the input file infile are assumed to be distributed as N(a+b*t,S^2) with unknown a, b and S^2. This operator estimates the parameter a and b. For every field element x only those timesteps t belong to the sample S(x), which have i(t,x) NE miss. Thus the estimation for a is stored in outfile1 and that for b is stored in outfile2. To subtract the trend from the data see operator subtrend. It is assumed that all timesteps are equidistant, if this is not the case set the parameter equal=false.

Usage

```
cdo_trend(ifile, equal = NULL, ofile1 = NULL, ofile2 = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`equal`	BOOL - Set to false for unequal distributed timesteps (default: true)
`ofile1, ofile2`	Strings with the path to the output files.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`trendarith` *Add or subtract a trend*

Description

This module is for adding or subtracting a trend computed by the operator trend.

Usage

```
cdo_addtrend(ifile1, ifile2, ifile3, equal = NULL, ofile = NULL)

cdo_subtrend(ifile1, ifile2, ifile3, equal = NULL, ofile = NULL)
```

Arguments

`ifile1, ifile2, ifile3`	Strings with the path to the input files.
`equal`	BOOL - Set to false for unequal distributed timesteps (default: true)
`ofile`	String with the path to the output file.

Details

```
addtrend  Add trend
          It is

          o(t,x) = i_1(t,x) + (i_2(1,x) + i_3(1,x)*t)
          where t is the timesteps.
subtrend  Subtract trend
          It is

          o(t,x) = i_1(t,x) - (i_2(1,x) + i_3(1,x)*t)
          where t is the timesteps.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`unpack` *Unpack data*

Description

Packing reduces the data volume by reducing the precision of the stored numbers. It is implemented using the NetCDF attributes add_offset and scale_factor. The operator unpack unpack all packed variables. The default data type for all variables is automatically changed to 32-bit floats. Use the CDO option -b F64 to change the data type to 64-bit floats, if needed.

Usage

```
cdo_unpack(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`vargen` *Generate a field*

Description

Generates a dataset with one or more fields

Usage

```
cdo_const(
  const = NULL,
  seed = NULL,
  grid = NULL,
  start = NULL,
  end = NULL,
  inc = NULL,
  levels = NULL,
  ofile = NULL
)

cdo_random(
  const = NULL,
  seed = NULL,
  grid = NULL,
  start = NULL,
  end = NULL,
  inc = NULL,
  levels = NULL,
  ofile = NULL
)

cdo_seq(
  const = NULL,
  seed = NULL,
  grid = NULL,
  start = NULL,
  end = NULL,
  inc = NULL,
  levels = NULL,
  ofile = NULL
)

cdo_stdatm(
  const = NULL,
  seed = NULL,
  grid = NULL,
  start = NULL,
  end = NULL,
  inc = NULL,
  levels = NULL,
  ofile = NULL
)

cdo_topo(
  const = NULL,
  seed = NULL,
  grid = NULL,
  start = NULL,
  end = NULL,
  inc = NULL,
  levels = NULL,
  ofile = NULL
)
```

Arguments

`const`	FLOAT - Constant
`seed`	INTEGER - The seed for a new sequence of pseudo-random numbers [default: 1]
`grid`	STRING - Target grid description file or name
`start`	FLOAT - Start value of the loop
`end`	FLOAT - End value of the loop
`inc`	FLOAT - Increment of the loop [default: 1]
`levels`	FLOAT - Target levels in metre above surface
`ofile`	String with the path to the output file.

Details

```
const   Create a constant field
      Creates a constant field. All field elements of the grid have the same value.
random  Create a field with random numbers
     Creates a field with rectangularly distrubuted random numbers in the interval \[0,1\].
topo    Create a field with topography
     Creates a field with topography data, per default on a global half degree grid.
seq     Create a time series
     Creates a time series with field size 1 and field elements beginning with a start value in time step
        which is increased from one time step to the next.
stdatm  Create values for pressure and temperature for hydrostatic atmosphere
     Creates pressure and temperature values for the given list of vertical levels.
        The formulas are:

     P(z) = P_0 * exp(-1 * g/R * H/T_0 * log( (exp(z/H)*T_0 + T_Delta)/(T_0 + T_Delta))
        T(z) = T_0 + T_Delta * exp(-z/H)

        with the following constants

        T_0     = 213 K           Offset to get a surface temperature of 288K
        T_Delta = 75 K            Temperature lapse rate for 10Km
        P_0     = 1013.25 hPa     Surface pressure
        H       = 10000.0 m       Scale height
        g       = 9.80665 m/s**2  Earth gravity
        R       = 287.05 J/kg*K   Gas constant for air
```

```
This is the solution for the hydrostatic equations and is only valid for the
troposphere (constant positive lapse rate). The temperature increase in the
 stratosphere and other effects of the upper atmosphere are not taken into
  account.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`varsstat` *Statistical values over all variables*

Description

This module computes statistical values over all variables for each timestep. Depending on the chosen operator the minimum, maximum, range, sum, average, variance or standard deviation is written to outfile. All input variables need to have the same gridsize and the same number of levels.

Usage

```
cdo_varsavg(ifile, ofile = NULL)

cdo_varsmax(ifile, ofile = NULL)

cdo_varsmean(ifile, ofile = NULL)

cdo_varsmin(ifile, ofile = NULL)

cdo_varsrange(ifile, ofile = NULL)

cdo_varsstd(ifile, ofile = NULL)

cdo_varsstd1(ifile, ofile = NULL)

cdo_varssum(ifile, ofile = NULL)

cdo_varsvar(ifile, ofile = NULL)

cdo_varsvar1(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Details

varsmin	Variables minimum For every timestep the minimum over all variables is computed.
varsmax	Variables maximum For every timestep the maximum over all variables is computed.
varsrange	Variables range For every timestep the range over all variables is computed.
varssum	Variables sum For every timestep the sum over all variables is computed.
varsmean	Variables mean For every timestep the mean over all variables is computed.
varsavg	Variables average For every timestep the average over all variables is computed.
varsstd	Variables standard deviation For every timestep the standard deviation over all variables is computed. Normalize by n.
varsstd1	Variables standard deviation (n-1) For every timestep the standard deviation over all variables is computed. Normalize by (n-1).
varsvar	Variables variance For every timestep the variance over all variables is computed. Normalize by n.
varsvar1	Variables variance (n-1) For every timestep the variance over all variables is computed. Normalize by (n-1).

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

---

verifygrid *Verify grid coordinates*

Description

This operator verifies the coordinates of all horizontal grids found in infile. Among other things, it searches for duplicate cells, non-convex cells, and whether the center is located outside the cell bounds. Use the CDO option -v to output the position of these cells. This information can be useful to avoid problems when interpolating the data.

Usage

```
cdo_verifygrid(ifile)
```

Arguments

ifile	String with the path to the input file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`vertfillmiss` *Vertical filling of missing values*

Description

This operator fills in vertical missing values. The method parameter can be used to select the filling method. The default method=nearest fills missing values with the nearest neighbor value. Other options are forward and backward to fill missing values by forward or backward propagation of values. Use the limit parameter to set the maximum number of consecutive missing values to fill and max_gaps to set the maximum number of gaps to fill.

Usage

```
cdo_vertfillmiss(
  ifile,
  method = NULL,
  limit = NULL,
  max_gaps = NULL,
  ofile = NULL
)
```

Arguments

`ifile`	String with the path to the input file.
`method`	STRING - Fill method [nearest|linear|forward|backward] (default: nearest)
`limit`	INTEGER - The maximum number of consecutive missing values to fill (default: all)
`max_gaps`	INTEGER - The maximum number of gaps to fill (default: all)
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

vertintap *Vertical pressure interpolation*

Description

Interpolate 3D variables on hybrid sigma height coordinates to pressure levels. The input file must contain the 3D air pressure in pascal. The air pressure is identified by the NetCDF CF standard name air_pressure. Use the alias ap2plx or the environment variable EXTRAPOLATE to extrapolate missing values. This operator requires all variables on the same horizontal grid.

Usage

```
cdo_ap2pl(ifile, plevels = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`plevels`	FLOAT - Comma-separated list of pressure levels in pascal
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

Note

This is a specific implementation for NetCDF files from the ICON model, it may not work with data from other sources.

vertintgh *Vertical height interpolation*

Description

Interpolate 3D variables on hybrid sigma height coordinates to height levels. The input file must contain the 3D geometric height in meter. The geometric height is identified by the NetCDF CF standard name geometric_height_at_full_level_center. Use the alias gh2hlx or the environment variable EXTRAPOLATE to extrapolate missing values. This operator requires all variables on the same horizontal grid.

Usage

```
cdo_gh2hl(ifile, hlevels = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`hlevels`	FLOAT - Comma-separated list of height levels in meter
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

Note

This is a specific implementation for NetCDF files from the ICON model, it may not work with data from other sources.

`vertintml` *Vertical interpolation*

Description

Interpolates 3D variables on hybrid sigma pressure level to pressure or height levels. To calculate the pressure on model levels, the a and b coefficients defining the model levels and the surface pressure are required. The a and b coefficients are normally part of the model level data. If not available, the surface pressure can be derived from the logarithm of the surface pressure. To extrapolate the temperature, the surface geopotential is also needed. The geopotential height must be present at the hybrid layer interfaces (model half-layers)! All needed variables are identified by their GRIB1 code number or NetCDF CF standard name. Supported parameter tables are: WMO standard table number 2 and ECMWF local table number 128. Name & Units & GRIB1 code & CF standard name log surface pressure & Pa & 152 & surface pressure & Pa & 134 & surface_air_pressure air temperature & K & 130 & air_temperature surface geopotential & m2 s-2 & 129 & surface_geopotential geopotential height & m & 156 & geopotential_height Use the alias ml2plx/ml2hlx or the environment variable EXTRAPOLATE to extrapolate missing values. This operator requires all variables on the same horizontal grid. Missing values in the input data are not supported.

Usage

```
cdo_ml2hl(ifile, plevels = NULL, hlevels = NULL, ofile = NULL)

cdo_ml2pl(ifile, plevels = NULL, hlevels = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`plevels`	FLOAT - Pressure levels in pascal
`hlevels`	FLOAT - Height levels in meter
`ofile`	String with the path to the output file.

Details

```
ml2pl  Model to pressure level interpolation
      Interpolates 3D variables on hybrid sigma pressure level to pressure level.
ml2hl  Model to height level interpolation
       Interpolates 3D variables on hybrid sigma pressure level to height level.
       The procedure is the same as for the operator ml2pl except for
       the pressure levels being calculated from the heights by:
       plevel = 101325*exp(hlevel/-7000)
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

Note

The components of the hybrid coordinate must always be avaiable at the hybrid layer interfaces even if the data is defined at the hybrid layer midpoints.

`vertstat` *Vertical statistics*

Description

This module computes statistical values over all levels of the input variables. According to chosen operator the vertical minimum, maximum, range, sum, average, variance or standard deviation is written to outfile.

Usage

```
cdo_vertavg(ifile, weights = NULL, ofile = NULL)

cdo_vertmax(ifile, weights = NULL, ofile = NULL)

cdo_vertmean(ifile, weights = NULL, ofile = NULL)

cdo_vertmin(ifile, weights = NULL, ofile = NULL)

cdo_vertrange(ifile, weights = NULL, ofile = NULL)

cdo_vertstd(ifile, weights = NULL, ofile = NULL)

cdo_vertstd1(ifile, weights = NULL, ofile = NULL)

cdo_vertsum(ifile, weights = NULL, ofile = NULL)
```

```
cdo_vertvar(ifile, weights = NULL, ofile = NULL)

cdo_vertvar1(ifile, weights = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`weights`	BOOL - weights=FALSE disables weighting by layer thickness [default: weights=TRUE]
`ofile`	String with the path to the output file.

Details

```
vertmin    Vertical minimum
           For every gridpoint the minimum over all levels is computed.
vertmax    Vertical maximum
           For every gridpoint the maximum over all levels is computed.
vertrange  Vertical range
           For every gridpoint the range over all levels is computed.
vertsum    Vertical sum
           For every gridpoint the sum over all levels is computed.
vertmean   Vertical mean
         For every gridpoint the layer weighted mean over all levels is computed.
vertavg    Vertical average
       For every gridpoint the layer weighted average over all levels is computed.
vertstd    Vertical standard deviation
       For every gridpoint the standard deviation over all levels is computed. Normalize by n.
vertstd1   Vertical standard deviation (n-1)
       For every gridpoint the standard deviation over all levels is computed. Normalize by (n-1).
vertvar    Vertical variance
       For every gridpoint the variance over all levels is computed. Normalize by n.
vertvar1   Vertical variance (n-1)
       For every gridpoint the variance over all levels is computed. Normalize by (n-1).
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`wct` *Windchill temperature*

Description

Let infile1 and infile2 be time series of temperature and wind speed fields, then a corresponding time series of resulting windchill temperatures is written to outfile. The wind chill temperature calculation is only valid for a temperature of T <= 33 °C and a wind speed of v >= 1.39 m/s. Whenever these conditions are not satisfied, a missing value is written to outfile. Note that temperature and wind speed fields have to be given in units of °C and m/s, respectively.

Usage

```
cdo_wct(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`wind` *Wind transformation*

Description

This module converts relative divergence and vorticity to U and V wind and vice versa. Divergence and vorticity are spherical harmonic coefficients in spectral space and U and V are on a global regular Gaussian grid. The Gaussian latitudes need to be ordered from north to south. Missing values are not supported. The relationship between the spectral resolution, governed by the truncation number T, and the grid resolution depends on the number of grid points at which the shortest wavelength field is represented. For a grid with 2N points between the poles (so 4N grid points in total around the globe) the relationship is: linear grid: the shortest wavelength is represented by 2 grid points $\rightarrow$ 4N $\simeq$ 2(TL + 1) quadratic grid: the shortest wavelength is represented by 3 grid points $\rightarrow$ 4N $\simeq$ 3(TQ + 1) cubic grid: the shortest wavelength is represented by 4 grid points $\rightarrow$ 4N $\simeq$ 4(TC + 1) The quadratic grid is used by ECHAM and ERA15. ERA40 is using a linear Gaussian grid reflected by the TL notation. The following table shows the calculation of the number of latitudes and the triangular truncation for the different grid types: Gridtype & Number of latitudes: nlat & Triangular truncation: ntr linear & NINT((ntr*2 + 1)/2) & (nlat*2 - 1) / 2 quadratic & NINT((ntr*3 + 1)/2) & (nlat*2 - 1) / 3 cubic & NINT((ntr*4 + 1)/2) & (nlat*2 - 1) / 4

Usage

```
cdo_dv2uv(ifile, gridtype = NULL, ofile = NULL)

cdo_uv2dv(ifile, gridtype = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`gridtype`	STRING - Type of the grid: quadratic, linear, cubic (default: quadratic)
`ofile`	String with the path to the output file.

Details

```
dv2uv  Divergence and vorticity to U and V wind
         Calculate U and V wind on a Gaussian grid from spherical harmonic
      coefficients of relative divergence and vorticity. The divergence and vorticity
         need to have the names sd and svo or code numbers 155 and 138.
uv2dv  U and V wind to divergence and vorticity
      Calculate spherical harmonic coefficients of relative divergence and vorticity
      from U and V wind. The U and V wind need to be on a Gaussian grid and need to have the
         names u and v or the code numbers 131 and 132.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

Note

To speed up the calculations, the Legendre polynoms are kept in memory. This requires a relatively large amount of memory. This is for example 12GB for T1279 data.

`wind2` *D and V to velocity potential and stream function*

Description

Calculate spherical harmonic coefficients of velocity potential and stream function from spherical harmonic coefficients of relative divergence and vorticity. The divergence and vorticity need to have the names sd and svo or code numbers 155 and 138.

Usage

```
cdo_dv2ps(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

---

xsinfo — *Extra short information*

Description

This module writes information about the structure of infiles to standard output. infiles is an arbitrary number of input files. All input files need to have the same structure with the same variables on different timesteps. The information displayed depends on the chosen operator.

Usage

```
cdo_xsinfo(ifiles)

cdo_xsinfop(ifiles)
```

Arguments

`ifiles`	Character vector with the path to the input files.

Details

```
xsinfo   Extra short information listed by parameter name
      Prints short information of a dataset. The information is divided into 4 sections.
         Section 1 prints one line per parameter with the following information:
         - institute and source
         - time c=constant v=varying
         - type of statistical processing
         - number of levels and z-axis number
         - horizontal grid size and number
         - data type
         - memory type (float or double)
         - parameter name
      Section 2 to 4 gives a short overview of all grid, vertical and time coordinates.
xsinfop  Extra short information listed by parameter identifier
      The same as operator xsinfo but using the identifier instead of the name to label the parameter.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

ydayarith *Multiyear daily arithmetic*

Description

This module performs simple arithmetic of a time series and one timestep with the same day of year. For each field in infile1 the corresponding field of the timestep in infile2 with the same day of year is used. The input files need to have the same structure with the same variables. Usually infile2 is generated by an operator of the module YDAYSTAT.

Usage

```
cdo_ydayadd(ifile1, ifile2, ofile = NULL)

cdo_ydaydiv(ifile1, ifile2, ofile = NULL)

cdo_ydaymul(ifile1, ifile2, ofile = NULL)

cdo_ydaysub(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Details

```
ydayadd  Add multi-year daily time series
         Adds a time series and a multi-year daily time series.
ydaysub  Subtract multi-year daily time series
         Subtracts a time series and a multi-year daily time series.
ydaymul  Multiply multi-year daily time series
         Multiplies a time series and a multi-year daily time series.
ydaydiv  Divide multi-year daily time series
         Divides a time series and a multi-year daily time series.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`ydaypctl` *Multiyear daily percentile values*

Description

This operator writes a certain percentile of each day of year in infile1 to outfile. The algorithm uses histograms with minimum and maximum bounds given in infile2 and infile3, respectively. The default number of histogram bins is 101. The default can be overridden by setting the environment variable CDO_PCTL_NBINS to a different value. The files infile2 and infile3 should be the result of corresponding ydaymin and ydaymax operations, respectively. The date information in an output field is the date of the last contributing input field. o(001,x) = pth percentile {i(t,x), day(i(t)) = 001} ... o(366,x) = pth percentile {i(t,x), day(i(t)) = 366}

Usage

```
cdo_ydaypctl(ifile1, ifile2, ifile3, p = NULL, ofile = NULL)
```

Arguments

`ifile1, ifile2, ifile3`	Strings with the path to the input files.
`p`	FLOAT - Percentile number in {0, ..., 100}
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`ydaystat` *Multiyear daily statistics*

Description

This module computes statistical values of each day of year. Depending on the chosen operator the minimum, maximum, range, sum, average, variance or standard deviation of each day of year in infile is written to outfile. The date information in an output field is the date of the last contributing input field.

Usage

```
cdo_ydayavg(ifile, ofile = NULL)

cdo_ydaymax(ifile, ofile = NULL)

cdo_ydaymean(ifile, ofile = NULL)

cdo_ydaymin(ifile, ofile = NULL)

cdo_ydayrange(ifile, ofile = NULL)

cdo_ydaystd(ifile, ofile = NULL)

cdo_ydaystd1(ifile, ofile = NULL)

cdo_ydaysum(ifile, ofile = NULL)

cdo_ydayvar(ifile, ofile = NULL)

cdo_ydayvar1(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Details

```
ydaymin    Multi-year daily minimum
           o(001,x) = min\{i(t,x), day(i(t)) = 001\}
                             ...
           o(366,x) = min\{i(t,x), day(i(t)) = 366\}
ydaymax    Multi-year daily maximum
           o(001,x) = max\{i(t,x), day(i(t)) = 001\}
                             ...
           o(366,x) = max\{i(t,x), day(i(t)) = 366\}
ydayrange  Multi-year daily range
           o(001,x) = range\{i(t,x), day(i(t)) = 001\}
                             ...
           o(366,x) = range\{i(t,x), day(i(t)) = 366\}
ydaysum    Multi-year daily sum
           o(001,x) = sum\{i(t,x), day(i(t)) = 001\}
                             ...
           o(366,x) = sum\{i(t,x), day(i(t)) = 366\}
ydaymean   Multi-year daily mean
           o(001,x) = mean\{i(t,x), day(i(t)) = 001\}
                             ...
           o(366,x) = mean\{i(t,x), day(i(t)) = 366\}
```

```
ydayavg    Multi-year daily average
           o(001,x) = avg\{i(t,x), day(i(t)) = 001\}
                            ...
           o(366,x) = avg\{i(t,x), day(i(t)) = 366\}
ydaystd    Multi-year daily standard deviation
           Normalize by n.

           o(001,x) = std\{i(t,x), day(i(t)) = 001\}
                            ...
           o(366,x) = std\{i(t,x), day(i(t)) = 366\}
ydaystd1   Multi-year daily standard deviation (n-1)
           Normalize by (n-1).

           o(001,x) = std1\{i(t,x), day(i(t)) = 001\}
                            ...
           o(366,x) = std1\{i(t,x), day(i(t)) = 366\}
ydayvar    Multi-year daily variance
           Normalize by n.

           o(001,x) = var\{i(t,x), day(i(t)) = 001\}
                            ...
           o(366,x) = var\{i(t,x), day(i(t)) = 366\}
ydayvar1   Multi-year daily variance (n-1)
           Normalize by (n-1).

           o(001,x) = var1\{i(t,x), day(i(t)) = 001\}
                            ...
           o(366,x) = var1\{i(t,x), day(i(t)) = 366\}
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

---

ydrunpctl *Multiyear daily running percentile values*

Description

This operator writes running percentile values for each day of year in infile1 to outfile. A certain percentile is computed for all timesteps in running windows of which the medium timestep corresponds to a certain day of year. The algorithm uses histograms with minimum and maximum bounds given in infile2 and infile3, respectively. The default number of histogram bins is 101. The default can be overridden by setting the environment variable CDO_PCTL_NBINS to a different value. The files infile2 and infile3 should be the result of corresponding ydrunmin and ydrunmax operations, respectively. The date information in an output field is the date of the timestep in the middle of the last contributing running window. Note that the operator have to be applied to a continuous time series of daily measurements in order to yield physically meaningful results. Also note that the output time series begins (nts-1)/2 timesteps after the first timestep of the input time series and ends (nts-1)/2 timesteps before the last. For input data which are complete but not continuous, such as time series of daily measurements for the same month or season within different years, the operator only yields physically meaningful results if the input time series does include the (nts-1)/2 days before and after each period of interest. o(001,x) = pth percentile {i(t,x), i(t+1,x), ..., i(t+nts-1,x); day[(i(t+(nts-1)/2)] = 001} ... o(366,x) = pth percentile {i(t,x), i(t+1,x), ..., i(t+nts-1,x); day[(i(t+(nts-1)/2)] = 366}

Usage

```
cdo_ydrunpctl(
  ifile1,
  ifile2,
  ifile3,
  p = NULL,
  nts = NULL,
  rm_c = NULL,
  pm_r8 = NULL,
  ofile = NULL
)
```

Arguments

`ifile1, ifile2, ifile3`	Strings with the path to the input files.
`p`	FLOAT - Percentile number in {0, ..., 100}
`nts`	INTEGER - Number of timesteps
`rm_c`	STRING - Read method circular
`pm_r8`	STRING - Percentile method rtype8
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don’t return filenames return a character vector with the string output.

ydrunstat	*Multiyear daily running statistics*

Description

This module writes running statistical values for each day of year in infile to outfile. Depending on the chosen operator, the minimum, maximum, sum, average, variance or standard deviation of all timesteps in running windows of which the medium timestep corresponds to a certain day of year is computed. The date information in an output field is the date of the timestep in the middle of the last contributing running window. Note that the operator have to be applied to a continuous time series of daily measurements in order to yield physically meaningful results. Also note that the output time series begins (nts-1)/2 timesteps after the first timestep of the input time series and ends (nts-1)/2 timesteps before the last one. For input data which are complete but not continuous, such as time series of daily measurements for the same month or season within different years, the operator yields physically meaningful results only if the input time series does include the (nts-1)/2 days before and after each period of interest.

Usage

```
cdo_ydrunavg(ifile, nts = NULL, rm_c = NULL, ofile = NULL)

cdo_ydrunmax(ifile, nts = NULL, rm_c = NULL, ofile = NULL)

cdo_ydrunmean(ifile, nts = NULL, rm_c = NULL, ofile = NULL)

cdo_ydrunmin(ifile, nts = NULL, rm_c = NULL, ofile = NULL)

cdo_ydrunstd(ifile, nts = NULL, rm_c = NULL, ofile = NULL)

cdo_ydrunstd1(ifile, nts = NULL, rm_c = NULL, ofile = NULL)

cdo_ydrunsum(ifile, nts = NULL, rm_c = NULL, ofile = NULL)

cdo_ydrunvar(ifile, nts = NULL, rm_c = NULL, ofile = NULL)

cdo_ydrunvar1(ifile, nts = NULL, rm_c = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`nts`	INTEGER - Number of timesteps
`rm_c`	STRING - Read method circular
`ofile`	String with the path to the output file.

Details

```
ydrunmin   Multi-year daily running minimum
       o(001,x) = min\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); day\[(i(t+(nts-1)/2)\] = 001\}
                                     ...
       o(366,x) = min\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); day\[(i(t+(nts-1)/2)\] = 366\}
ydrunmax   Multi-year daily running maximum
       o(001,x) = max\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); day\[(i(t+(nts-1)/2)\] = 001\}
                                     ...
       o(366,x) = max\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); day\[(i(t+(nts-1)/2)\] = 366\}
ydrunsum   Multi-year daily running sum
       o(001,x) = sum\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); day\[(i(t+(nts-1)/2)\] = 001\}
                                     ...
       o(366,x) = sum\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); day\[(i(t+(nts-1)/2)\] = 366\}
ydrunmean  Multi-year daily running mean
       o(001,x) = mean\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); day\[(i(t+(nts-1)/2)\] = 001\}
                                     ...
       o(366,x) = mean\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); day\[(i(t+(nts-1)/2)\] = 366\}
ydrunavg   Multi-year daily running average
       o(001,x) = avg\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); day\[(i(t+(nts-1)/2)\] = 001\}
                                     ...
       o(366,x) = avg\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); day\[(i(t+(nts-1)/2)\] = 366\}
ydrunstd   Multi-year daily running standard deviation
           Normalize by n.

       o(001,x) = std\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); day\[i(t+(nts-1)/2)\] = 001\}
                                     ...
       o(366,x) = std\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); day\[i(t+(nts-1)/2)\] = 366\}
ydrunstd1  Multi-year daily running standard deviation (n-1)
           Normalize by (n-1).

       o(001,x) = std1\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); day\[i(t+(nts-1)/2)\] = 001\}
                                     ...
       o(366,x) = std1\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); day\[i(t+(nts-1)/2)\] = 366\}
ydrunvar   Multi-year daily running variance
           Normalize by n.

       o(001,x) = var\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); day\[(i(t+(nts-1)/2)\] = 001\}
                                     ...
       o(366,x) = var\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); day\[(i(t+(nts-1)/2)\] = 366\}
ydrunvar1  Multi-year daily running variance (n-1)
           Normalize by (n-1).

       o(001,x) = var1\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); day\[(i(t+(nts-1)/2)\] = 001\}
                                     ...
       o(366,x) = var1\{i(t,x), i(t+1,x), ..., i(t+nts-1,x); day\[(i(t+(nts-1)/2)\] = 366\}
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`yeararith` *Yearly arithmetic*

Description

This module performs simple arithmetic of a time series and one timestep with the same year. For each field in infile1 the corresponding field of the timestep in infile2 with the same year is used. The header information in infile1 have to be the same as in infile2. Usually infile2 is generated by an operator of the module YEARSTAT.

Usage

```
cdo_yearadd(ifile1, ifile2, ofile = NULL)

cdo_yeardiv(ifile1, ifile2, ofile = NULL)

cdo_yearmul(ifile1, ifile2, ofile = NULL)

cdo_yearsub(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Details

```
yearadd  Add yearly time series
         Adds a time series and a yearly time series.
yearsub  Subtract yearly time series
         Subtracts a time series and a yearly time series.
yearmul  Multiply yearly time series
         Multiplies a time series and a yearly time series.
yeardiv  Divide yearly time series
         Divides a time series and a yearly time series.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`yearmonstat` *Yearly mean from monthly data*

Description

This operator computes the yearly mean of a monthly time series. Each month is weighted with the number of days per month. The time of outfile is determined by the time in the middle of all contributing timesteps of infile. For every adjacent sequence t_1, ...,t_n of timesteps of the same year it is: o(t,x) = mean{i(t',x), t_1<t'<=t_n}

Usage

```
cdo_yearmonmean(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`yearpctl` *Yearly percentile values*

Description

This operator computes percentiles over all timesteps of the same year in infile1. The algorithm uses histograms with minimum and maximum bounds given in infile2 and infile3, respectively. The default number of histogram bins is 101. The default can be overridden by defining the environment variable CDO_PCTL_NBINS. The files infile2 and infile3 should be the result of corresponding yearmin and yearmax operations, respectively. The time of outfile is determined by the time in the middle of all contributing timesteps of infile1. This can be change with the CDO option – timestat_date <first|middle|last>. For every adjacent sequence t_1, ...,t_n of timesteps of the same year it is: o(t,x) = pth percentile {i(t',x), t_1<t'<=t_n}

Usage

```
cdo_yearpctl(ifile1, ifile2, ifile3, p = NULL, ofile = NULL)
```

Arguments

`ifile1, ifile2, ifile3`	Strings with the path to the input files.
`p`	FLOAT - Percentile number in {0, ..., 100}
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`yearstat` *Yearly statistics*

Description

This module computes statistical values over timesteps of the same year. Depending on the chosen operator the minimum, maximum, range, sum, average, variance or standard deviation of timesteps of the same year is written to outfile. The time of outfile is determined by the time in the middle of all contributing timesteps of infile. This can be change with the CDO option –timestat_date <first|middle|last>.

Usage

```
cdo_yearavg(ifile, complete_only = NULL, ofile = NULL)

cdo_yearmax(ifile, complete_only = NULL, ofile = NULL)

cdo_yearmaxidx(ifile, complete_only = NULL, ofile = NULL)

cdo_yearmean(ifile, complete_only = NULL, ofile = NULL)

cdo_yearmin(ifile, complete_only = NULL, ofile = NULL)

cdo_yearminidx(ifile, complete_only = NULL, ofile = NULL)

cdo_yearrange(ifile, complete_only = NULL, ofile = NULL)

cdo_yearstd(ifile, complete_only = NULL, ofile = NULL)

cdo_yearstd1(ifile, complete_only = NULL, ofile = NULL)

cdo_yearsum(ifile, complete_only = NULL, ofile = NULL)
```

```
cdo_yearvar(ifile, complete_only = NULL, ofile = NULL)

cdo_yearvar1(ifile, complete_only = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`complete_only`	BOOL - Process the last year only if it is complete
`ofile`	String with the path to the output file.

Details

```
yearmin     Yearly minimum
        For every adjacent sequence t_1, ...,t_n of timesteps of the same year it is:

            o(t,x) = min\{i(t',x), t_1&lt;t'&lt;=t_n\}
yearmax     Yearly maximum
        For every adjacent sequence t_1, ...,t_n of timesteps of the same year it is:

            o(t,x) = max\{i(t',x), t_1&lt;t'&lt;=t_n\}
yearminidx  Index of yearly minimum
        For every adjacent sequence t_1, ...,t_n of timesteps of the same year it is:

            o(t,x) = minidx\{i(t',x), t_1&lt;t'&lt;=t_n\}
yearmaxidx  Index of yearly maximum
        For every adjacent sequence t_1, ...,t_n of timesteps of the same year it is:

            o(t,x) = maxidx\{i(t',x), t_1&lt;t'&lt;=t_n\}
yearrange   Yearly range
        For every adjacent sequence t_1, ...,t_n of timesteps of the same year it is:

            o(t,x) = range\{i(t',x), t_1&lt;t'&lt;=t_n\}
yearsum     Yearly sum
        For every adjacent sequence t_1, ...,t_n of timesteps of the same year it is:

            o(t,x) = sum\{i(t',x), t_1&lt;t'&lt;=t_n\}
yearmean    Yearly mean
        For every adjacent sequence t_1, ...,t_n of timesteps of the same year it is:

            o(t,x) = mean\{i(t',x), t_1&lt;t'&lt;=t_n\}
yearavg     Yearly average
        For every adjacent sequence t_1, ...,t_n of timesteps of the same year it is:

            o(t,x) = avg\{i(t',x), t_1&lt;t'&lt;=t_n\}
yearstd     Yearly standard deviation
        Normalize by n. For every adjacent sequence t_1, ...,t_n of timesteps of the same year it is:

            o(t,x) = std\{i(t',x), t_1 &lt; t' &lt;= t_n\}
```

```
yearstd1    Yearly standard deviation (n-1)
        Normalize by (n-1). For every adjacent sequence t_1, ...,t_n of timesteps of the same year it is

            o(t,x) = std1\{i(t',x), t_1 &lt; t' &lt;= t_n\}
yearvar     Yearly variance
        Normalize by n. For every adjacent sequence t_1, ...,t_n of timesteps of the same year it is:

            o(t,x) = var\{i(t',x), t_1 &lt; t' &lt;= t_n\}
yearvar1    Yearly variance (n-1)
        Normalize by (n-1). For every adjacent sequence t_1, ...,t_n of timesteps of the same year it is

            o(t,x) = var1\{i(t',x), t_1 &lt; t' &lt;= t_n\}
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

Note

The operators yearmean and yearavg compute only arithmetical means!

`yhourarith` *Multiyear hourly arithmetic*

Description

This module performs simple arithmetic of a time series and one timestep with the same hour and day of year. For each field in infile1 the corresponding field of the timestep in infile2 with the same hour and day of year is used. The input files need to have the same structure with the same variables. Usually infile2 is generated by an operator of the module YHOURSTAT.

Usage

```
cdo_yhouradd(ifile1, ifile2, ofile = NULL)

cdo_yhourdiv(ifile1, ifile2, ofile = NULL)

cdo_yhourmul(ifile1, ifile2, ofile = NULL)

cdo_yhoursub(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Details

yhouradd	Add multi-year hourly time series Adds a time series and a multi-year hourly time series.
yhoursub	Subtract multi-year hourly time series Subtracts a time series and a multi-year hourly time series.
yhourmul	Multiply multi-year hourly time series Multiplies a time series and a multi-year hourly time series.
yhourdiv	Divide multi-year hourly time series Divides a time series and a multi-year hourly time series.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

---

yhourstat *Multiyear hourly statistics*

Description

This module computes statistical values of each hour and day of year. Depending on the chosen operator the minimum, maximum, range, sum, average, variance or standard deviation of each hour and day of year in infile is written to outfile. The date information in an output field is the date of the last contributing input field.

Usage

```
cdo_yhouravg(ifile, ofile = NULL)

cdo_yhourmax(ifile, ofile = NULL)

cdo_yhourmean(ifile, ofile = NULL)

cdo_yhourmin(ifile, ofile = NULL)

cdo_yhourrange(ifile, ofile = NULL)

cdo_yhourstd(ifile, ofile = NULL)

cdo_yhourstd1(ifile, ofile = NULL)

cdo_yhoursum(ifile, ofile = NULL)

cdo_yhourvar(ifile, ofile = NULL)

cdo_yhourvar1(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Details

```
yhourmin    Multi-year hourly minimum
            o(0001,x) = min\{i(t,x), day(i(t)) = 0001\}
                                ...
            o(8784,x) = min\{i(t,x), day(i(t)) = 8784\}
yhourmax    Multi-year hourly maximum
            o(0001,x) = max\{i(t,x), day(i(t)) = 0001\}
                                ...
            o(8784,x) = max\{i(t,x), day(i(t)) = 8784\}
yhourrange  Multi-year hourly range
            o(0001,x) = range\{i(t,x), day(i(t)) = 0001\}
                                ...
            o(8784,x) = range\{i(t,x), day(i(t)) = 8784\}
yhoursum    Multi-year hourly sum
            o(0001,x) = sum\{i(t,x), day(i(t)) = 0001\}
                                ...
            o(8784,x) = sum\{i(t,x), day(i(t)) = 8784\}
yhourmean   Multi-year hourly mean
            o(0001,x) = mean\{i(t,x), day(i(t)) = 0001\}
                                ...
            o(8784,x) = mean\{i(t,x), day(i(t)) = 8784\}
yhouravg    Multi-year hourly average
            o(0001,x) = avg\{i(t,x), day(i(t)) = 0001\}
                                ...
            o(8784,x) = avg\{i(t,x), day(i(t)) = 8784\}
yhourstd    Multi-year hourly standard deviation
            Normalize by n.

            o(0001,x) = std\{i(t,x), day(i(t)) = 0001\}
                                ...
            o(8784,x) = std\{i(t,x), day(i(t)) = 8784\}
yhourstd1   Multi-year hourly standard deviation (n-1)
            Normalize by (n-1).

            o(0001,x) = std1\{i(t,x), day(i(t)) = 0001\}
                                ...
            o(8784,x) = std1\{i(t,x), day(i(t)) = 8784\}
yhourvar    Multi-year hourly variance
            Normalize by n.

            o(0001,x) = var\{i(t,x), day(i(t)) = 0001\}
                                ...
            o(8784,x) = var\{i(t,x), day(i(t)) = 8784\}
```

```
yhourvar1   Multi-year hourly variance (n-1)
            Normalize by (n-1).

            o(0001,x) = var1\{i(t,x), day(i(t)) = 0001\}
                                ...
            o(8784,x) = var1\{i(t,x), day(i(t)) = 8784\}
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

---

ymonarith *Multiyear monthly arithmetic*

Description

This module performs simple arithmetic of a time series and one timestep with the same month of year. For each field in infile1 the corresponding field of the timestep in infile2 with the same month of year is used. The input files need to have the same structure with the same variables. Usually infile2 is generated by an operator of the module YMONSTAT.

Usage

```
cdo_ymonadd(ifile1, ifile2, ofile = NULL)

cdo_ymondiv(ifile1, ifile2, ofile = NULL)

cdo_ymonmul(ifile1, ifile2, ofile = NULL)

cdo_ymonsub(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Details

```
ymonadd  Add multi-year monthly time series
         Adds a time series and a multi-year monthly time series.
ymonsub  Subtract multi-year monthly time series
         Subtracts a time series and a multi-year monthly time series.
ymonmul  Multiply multi-year monthly time series
         Multiplies a time series with a multi-year monthly time series.
ymondiv  Divide multi-year monthly time series
         Divides a time series by a multi-year monthly time series.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

---

ymoncomp *Multiyear monthly comparison*

Description

This module performs compaisons of a time series and one timestep with the same month of year. For each field in infile1 the corresponding field of the timestep in infile2 with the same month of year is used. The resulting field is a mask containing 1 if the comparison is true and 0 if not. The type of comparison depends on the chosen operator. The input files need to have the same structure with the same variables. Usually infile2 is generated by an operator of the module YMONSTAT.

Usage

```
cdo_ymoneq(ifile1, ifile2, ofile = NULL)

cdo_ymonge(ifile1, ifile2, ofile = NULL)

cdo_ymongt(ifile1, ifile2, ofile = NULL)

cdo_ymonle(ifile1, ifile2, ofile = NULL)

cdo_ymonlt(ifile1, ifile2, ofile = NULL)

cdo_ymonne(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Details

```
ymoneq  Compare time series with Equal
      Compares whether a time series is equal to a multi-year monthly time series.
ymonne  Compare time series with NotEqual
     Compares whether a time series is not equal to a multi-year monthly time series.
ymonle  Compare time series with LessEqual
     Compares whether a time series is less than or equal to a multi-year monthly time series.
ymonlt  Compares if time series with LessThan
     Compares whether a time series is less than a multi-year monthly time series.
ymonge  Compares if time series with GreaterEqual
```

```
     Compares whether a time series is greater than or equal to a multi-year monthly time series.
ymongt  Compares if time series with GreaterThan
     Compares whether a time series is greater than a multi-year monthly time series.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`ymonpctl` *Multiyear monthly percentile values*

Description

This operator writes a certain percentile of each month of year in infile1 to outfile. The algorithm uses histograms with minimum and maximum bounds given in infile2 and infile3, respectively. The default number of histogram bins is 101. The default can be overridden by setting the environment variable CDO_PCTL_NBINS to a different value. The files infile2 and infile3 should be the result of corresponding ymonmin and ymonmax operations, respectively. The date information in an output field is the date of the last contributing input field. o(01,x) = pth percentile {i(t,x), month(i(t)) = 01} ... o(12,x) = pth percentile {i(t,x), month(i(t)) = 12}

Usage

```
cdo_ymonpctl(ifile1, ifile2, ifile3, p = NULL, ofile = NULL)
```

Arguments

`ifile1, ifile2, ifile3`	Strings with the path to the input files.
`p`	FLOAT - Percentile number in {0, ..., 100}
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`ymonstat` *Multiyear monthly statistics*

Description

This module computes statistical values of each month of year. Depending on the chosen operator the minimum, maximum, range, sum, average, variance or standard deviation of each month of year in infile is written to outfile. The date information in an output field is the date of the last contributing input field. This can be change with the CDO option –timestat_date <first|middle|last>.

Usage

```
cdo_ymonavg(ifile, ofile = NULL)

cdo_ymonmax(ifile, ofile = NULL)

cdo_ymonmean(ifile, ofile = NULL)

cdo_ymonmin(ifile, ofile = NULL)

cdo_ymonrange(ifile, ofile = NULL)

cdo_ymonstd(ifile, ofile = NULL)

cdo_ymonstd1(ifile, ofile = NULL)

cdo_ymonsum(ifile, ofile = NULL)

cdo_ymonvar(ifile, ofile = NULL)

cdo_ymonvar1(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Details

```
ymonmin    Multi-year monthly minimum
           o(01,x) = min\{i(t,x), month(i(t)) = 01\}
                                ...
           o(12,x) = min\{i(t,x), month(i(t)) = 12\}
ymonmax    Multi-year monthly maximum
           o(01,x) = max\{i(t,x), month(i(t)) = 01\}
                                ...
           o(12,x) = max\{i(t,x), month(i(t)) = 12\}
```

```
ymonrange  Multi-year monthly range
           o(01,x) = range\{i(t,x), month(i(t)) = 01\}
                            ...
           o(12,x) = range\{i(t,x), month(i(t)) = 12\}
ymonsum    Multi-year monthly sum
           o(01,x) = sum\{i(t,x), month(i(t)) = 01\}
                            ...
           o(12,x) = sum\{i(t,x), month(i(t)) = 12\}
ymonmean   Multi-year monthly mean
           o(01,x) = mean\{i(t,x), month(i(t)) = 01\}
                            ...
           o(12,x) = mean\{i(t,x), month(i(t)) = 12\}
ymonavg    Multi-year monthly average
           o(01,x) = avg\{i(t,x), month(i(t)) = 01\}
                            ...
           o(12,x) = avg\{i(t,x), month(i(t)) = 12\}
ymonstd    Multi-year monthly standard deviation
           Normalize by n.

           o(01,x) = std\{i(t,x), month(i(t)) = 01\}
                            ...
           o(12,x) = std\{i(t,x), month(i(t)) = 12\}
ymonstd1   Multi-year monthly standard deviation (n-1)
           Normalize by (n-1).

           o(01,x) = std1\{i(t,x), month(i(t)) = 01\}
                            ...
           o(12,x) = std1\{i(t,x), month(i(t)) = 12\}
ymonvar    Multi-year monthly variance
           Normalize by n.

           o(01,x) = var\{i(t,x), month(i(t)) = 01\}
                            ...
           o(12,x) = var\{i(t,x), month(i(t)) = 12\}
ymonvar1   Multi-year monthly variance (n-1)
           Normalize by (n-1).

           o(01,x) = var1\{i(t,x), month(i(t)) = 01\}
                            ...
           o(12,x) = var1\{i(t,x), month(i(t)) = 12\}
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

yseasarith *Multiyear seasonal arithmetic*

Description

This module performs simple arithmetic of a time series and one timestep with the same season. For each field in infile1 the corresponding field of the timestep in infile2 with the same season is used. The input files need to have the same structure with the same variables. Usually infile2 is generated by an operator of the module YSEASSTAT.

Usage

```
cdo_yseasadd(ifile1, ifile2, ofile = NULL)

cdo_yseasdiv(ifile1, ifile2, ofile = NULL)

cdo_yseasmul(ifile1, ifile2, ofile = NULL)

cdo_yseassub(ifile1, ifile2, ofile = NULL)
```

Arguments

`ifile1, ifile2`	Strings with the path to the input files.
`ofile`	String with the path to the output file.

Details

```
yseasadd  Add multi-year seasonal time series
          Adds a time series and a multi-year seasonal time series.
yseassub  Subtract multi-year seasonal time series
          Subtracts a time series and a multi-year seasonal time series.
yseasmul  Multiply multi-year seasonal time series
          Multiplies a time series and a multi-year seasonal time series.
yseasdiv  Divide multi-year seasonal time series
          Divides a time series and a multi-year seasonal time series.
```

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`yseaspctl` *Multiyear seasonal percentile values*

Description

This operator writes a certain percentile of each season in infile1 to outfile. The algorithm uses histograms with minimum and maximum bounds given in infile2 and infile3, respectively. The default number of histogram bins is 101. The default can be overridden by setting the environment variable CDO_PCTL_NBINS to a different value. The files infile2 and infile3 should be the result of corresponding yseasmin and yseasmax operations, respectively. The date information in an output field is the date of the last contributing input field. o(1,x) = pth percentile {i(t,x), month(i(t)) = 12, 01, 02} o(2,x) = pth percentile {i(t,x), month(i(t)) = 03, 04, 05} o(3,x) = pth percentile {i(t,x), month(i(t)) = 06, 07, 08} o(4,x) = pth percentile {i(t,x), month(i(t)) = 09, 10, 11}

Usage

```
cdo_yseaspctl(ifile1, ifile2, ifile3, p = NULL, ofile = NULL)
```

Arguments

`ifile1, ifile2, ifile3`	Strings with the path to the input files.
`p`	FLOAT - Percentile number in {0, ..., 100}
`ofile`	String with the path to the output file.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don’t return filenames return a character vector with the string output.

`yseasstat` *Multiyear seasonal statistics*

Description

This module computes statistical values of each season. Depending on the chosen operator the minimum, maximum, range, sum, average, variance or standard deviation of each season in infile is written to outfile. The date information in an output field is the date of the last contributing input field.

Usage

```
cdo_yseasavg(ifile, ofile = NULL)

cdo_yseasmax(ifile, ofile = NULL)

cdo_yseasmean(ifile, ofile = NULL)

cdo_yseasmin(ifile, ofile = NULL)

cdo_yseasrange(ifile, ofile = NULL)

cdo_yseasstd(ifile, ofile = NULL)

cdo_yseasstd1(ifile, ofile = NULL)

cdo_yseassum(ifile, ofile = NULL)

cdo_yseasvar(ifile, ofile = NULL)

cdo_yseasvar1(ifile, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`ofile`	String with the path to the output file.

Details

```
yseasmin    Multi-year seasonal minimum
            o(1,x) = min\{i(t,x), month(i(t)) = 12, 01, 02\}
            o(2,x) = min\{i(t,x), month(i(t)) = 03, 04, 05\}
            o(3,x) = min\{i(t,x), month(i(t)) = 06, 07, 08\}
            o(4,x) = min\{i(t,x), month(i(t)) = 09, 10, 11\}
yseasmax    Multi-year seasonal maximum
            o(1,x) = max\{i(t,x), month(i(t)) = 12, 01, 02\}
            o(2,x) = max\{i(t,x), month(i(t)) = 03, 04, 05\}
            o(3,x) = max\{i(t,x), month(i(t)) = 06, 07, 08\}
            o(4,x) = max\{i(t,x), month(i(t)) = 09, 10, 11\}
yseasrange  Multi-year seasonal range
            o(1,x) = range\{i(t,x), month(i(t)) = 12, 01, 02\}
            o(2,x) = range\{i(t,x), month(i(t)) = 03, 04, 05\}
            o(3,x) = range\{i(t,x), month(i(t)) = 06, 07, 08\}
            o(4,x) = range\{i(t,x), month(i(t)) = 09, 10, 11\}
yseassum    Multi-year seasonal sum
            o(1,x) = sum\{i(t,x), month(i(t)) = 12, 01, 02\}
            o(2,x) = sum\{i(t,x), month(i(t)) = 03, 04, 05\}
            o(3,x) = sum\{i(t,x), month(i(t)) = 06, 07, 08\}
            o(4,x) = sum\{i(t,x), month(i(t)) = 09, 10, 11\}
```

`yseasmean`	`Multi-year seasonal mean`
	`o(1,x) = mean\{i(t,x), month(i(t)) = 12, 01, 02\}`
	`o(2,x) = mean\{i(t,x), month(i(t)) = 03, 04, 05\}`
	`o(3,x) = mean\{i(t,x), month(i(t)) = 06, 07, 08\}`
	`o(4,x) = mean\{i(t,x), month(i(t)) = 09, 10, 11\}`

`yseasavg`	`Multi-year seasonal average`
	`o(1,x) = avg\{i(t,x), month(i(t)) = 12, 01, 02\}`
	`o(2,x) = avg\{i(t,x), month(i(t)) = 03, 04, 05\}`
	`o(3,x) = avg\{i(t,x), month(i(t)) = 06, 07, 08\}`
	`o(4,x) = avg\{i(t,x), month(i(t)) = 09, 10, 11\}`

`yseasstd`	`Multi-year seasonal standard deviation`
	`o(1,x) = std\{i(t,x), month(i(t)) = 12, 01, 02\}`
	`o(2,x) = std\{i(t,x), month(i(t)) = 03, 04, 05\}`
	`o(3,x) = std\{i(t,x), month(i(t)) = 06, 07, 08\}`
	`o(4,x) = std\{i(t,x), month(i(t)) = 09, 10, 11\}`

`yseasstd1`	`Multi-year seasonal standard deviation (n-1)`
	`o(1,x) = std1\{i(t,x), month(i(t)) = 12, 01, 02\}`
	`o(2,x) = std1\{i(t,x), month(i(t)) = 03, 04, 05\}`
	`o(3,x) = std1\{i(t,x), month(i(t)) = 06, 07, 08\}`
	`o(4,x) = std1\{i(t,x), month(i(t)) = 09, 10, 11\}`

`yseasvar`	`Multi-year seasonal variance`
	`o(1,x) = var\{i(t,x), month(i(t)) = 12, 01, 02\}`
	`o(2,x) = var\{i(t,x), month(i(t)) = 03, 04, 05\}`
	`o(3,x) = var\{i(t,x), month(i(t)) = 06, 07, 08\}`
	`o(4,x) = var\{i(t,x), month(i(t)) = 09, 10, 11\}`

`yseasvar1`	`Multi-year seasonal variance (n-1)`
	`o(1,x) = var1\{i(t,x), month(i(t)) = 12, 01, 02\}`
	`o(2,x) = var1\{i(t,x), month(i(t)) = 03, 04, 05\}`
	`o(3,x) = var1\{i(t,x), month(i(t)) = 06, 07, 08\}`
	`o(4,x) = var1\{i(t,x), month(i(t)) = 09, 10, 11\}`

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

`zonstat` *Zonal statistics*

Description

This module computes zonal statistical values of the input fields. Depending on the chosen operator, the zonal minimum, maximum, range, sum, average, standard deviation, variance, skewness, kurtosis, median or a certain percentile of the field is written to outfile. Operators of this module require all variables on the same regular lon/lat grid. Only the zonal mean (zonmean) can be calculated for data on an unstructured grid if the latitude bins are defined with the optional parameter zonaldes.

Usage

```
cdo_zonavg(ifile, p = NULL, zonaldes = NULL, ofile = NULL)

cdo_zonkurt(ifile, p = NULL, zonaldes = NULL, ofile = NULL)

cdo_zonmax(ifile, p = NULL, zonaldes = NULL, ofile = NULL)

cdo_zonmean(ifile, p = NULL, zonaldes = NULL, ofile = NULL)

cdo_zonmedian(ifile, p = NULL, zonaldes = NULL, ofile = NULL)

cdo_zonmin(ifile, p = NULL, zonaldes = NULL, ofile = NULL)

cdo_zonpctl(ifile, p = NULL, zonaldes = NULL, ofile = NULL)

cdo_zonrange(ifile, p = NULL, zonaldes = NULL, ofile = NULL)

cdo_zonskew(ifile, p = NULL, zonaldes = NULL, ofile = NULL)

cdo_zonstd(ifile, p = NULL, zonaldes = NULL, ofile = NULL)

cdo_zonstd1(ifile, p = NULL, zonaldes = NULL, ofile = NULL)

cdo_zonsum(ifile, p = NULL, zonaldes = NULL, ofile = NULL)

cdo_zonvar(ifile, p = NULL, zonaldes = NULL, ofile = NULL)

cdo_zonvar1(ifile, p = NULL, zonaldes = NULL, ofile = NULL)
```

Arguments

`ifile`	String with the path to the input file.
`p`	FLOAT - Percentile number in {0, ..., 100}
`zonaldes`	STRING - Description of the zonal latitude bins needed for data on an unstructured grid. A predefined zonal description is zonal_<DY>. DY is the increment of the latitudes in degrees.
`ofile`	String with the path to the output file.

Details

```
zonmin     Zonal minimum
           For every latitude the minimum over all longitudes is computed.
zonmax     Zonal maximum
           For every latitude the maximum over all longitudes is computed.
zonrange   Zonal range
           For every latitude the range over all longitudes is computed.
zonsum     Zonal sum
```

	For every latitude the sum over all longitudes is computed.
zonmean	Zonal mean For every latitude the mean over all longitudes is computed. Use the optional parameter zonaldes for data on an unstructured grid.
zonavg	Zonal average For every latitude the average over all longitudes is computed.
zonstd	Zonal standard deviation For every latitude the standard deviation over all longitudes is computed. Normalize by n.
zonstd1	Zonal standard deviation (n-1) For every latitude the standard deviation over all longitudes is computed. Normalize by (n-1).
zonvar	Zonal variance For every latitude the variance over all longitudes is computed. Normalize by n.
zonvar1	Zonal variance (n-1) For every latitude the variance over all longitudes is computed. Normalize by (n-1).
zonskew	Zonal skewness For every latitude the skewness over all longitudes is computed.
zonkurt	Zonal kurtosis For every latitude the kurtosis over all longitudes is computed.
zonmedian	Zonal median For every latitude the median over all longitudes is computed.
zonpctl	Zonal percentiles For every latitude the pth percentile over all longitudes is computed.

Value

Operators that output one or more files return a character vector to the output files.

Operators that output an indefinite number of files return a string with the basename of the files.

Operators that don't return filenames return a character vector with the string output.

Index