

# CDO User's Guide

---

Climate Data Operators  
Version 1.1.1  
April 2008

Uwe Schulzweida, Luis Kornblueh – *MPI for Meteorology*  
Ralf Quast – *Brockmann Consult*

---

# Contents

<b>1. Introduction</b>	<b>6</b>
1.1. Building from sources	6
1.1.1. Compilation	7
1.1.2. Installation	7
1.2. Usage	7
1.2.1. Options	8
1.2.2. Operators	8
1.2.3. Combining operators	8
1.2.4. Operator parameter	9
1.3. Grid description	9
1.3.1. Predefined grids	9
1.3.2. Grids from data files	10
1.3.3. SCRIP grids	10
1.3.4. PINGO grids	10
1.3.5. CDO grids	11
1.4. Z-axis description	12
1.5. Time axis	13
1.5.1. Absolute time	13
1.5.2. Relative time	13
1.5.3. Conversion of the time	13
1.6. Parameter table	13
1.7. Missing values	14
1.7.1. Mean and average	15
<b>2. Reference manual</b>	<b>16</b>
2.1. Information	17
2.1.1. INFO - Information and simple statistics	18
2.1.2. SINFO - Short information	19
2.1.3. DIFF - Compare two datasets field by field	20
2.1.4. NINFO - Print the number of parameters, levels or times	21
2.1.5. SHOWINFO - Show variables, levels or times	22
2.1.6. FILEDES - Dataset description	23
2.2. File operations	24
2.2.1. COPY - Copy datasets	25
2.2.2. REPLACE - Replace variables	25
2.2.3. MERGE - Merge datasets	26
2.2.4. SPLIT - Split a dataset	27
2.2.5. SPLITTIME - Split time steps of a dataset	28
2.2.6. SPLITSEL - Split selected time steps	29
2.3. Selection	30
2.3.1. SELECT - Select fields	31
2.3.2. SELTIME - Select time steps	33
2.3.3. SELBOX - Select a box of a field	35
2.4. Conditional selection	36
2.4.1. COND - Conditional select one field	37
2.4.2. COND2 - Conditional select two fields	37
2.4.3. CONDC - Conditional select a constant	38
2.5. Comparison	39
2.5.1. COMP - Comparison of two fields	40

2.5.2.	COMPC - Comparison of a field with a constant	41
2.6.	Modification	42
2.6.1.	SET - Set field info	43
2.6.2.	SETTIME - Set time	44
2.6.3.	CHANGE - Change field header	46
2.6.4.	SETGRID - Set grid type	47
2.6.5.	SETZAXIS - Set zaxis type	47
2.6.6.	SETGATT - Set global attribute	48
2.6.7.	INVERT - Invert fields	49
2.6.8.	MASKREGION - Mask regions	50
2.6.9.	MASKBOX - Mask a box	51
2.6.10.	SETBOX - Set a box to constant	52
2.6.11.	ENLARGE - Enlarge fields	53
2.6.12.	SETMISS - Set missing value	54
2.7.	Arithmetic	55
2.7.1.	EXPR - Evaluate expressions	56
2.7.2.	MATH - Mathematical functions	57
2.7.3.	ARITHC - Arithmetic with a constant	58
2.7.4.	ARITH - Arithmetic on two datasets	59
2.7.5.	MONARITH - Monthly arithmetic	60
2.7.6.	YMONARITH - Multi-year monthly arithmetic	61
2.7.7.	ARITHDAYS - Arithmetic with days	62
2.8.	Statistical values	63
2.8.1.	ENSSTAT - Statistical values over an ensemble	67
2.8.2.	FLDSTAT - Statistical values over a field	69
2.8.3.	ZONSTAT - Zonal statistical values	71
2.8.4.	MERSTAT - Meridional statistical values	73
2.8.5.	VERTSTAT - Vertical statistical values	75
2.8.6.	TIMSELSTAT - Time range statistical values	76
2.8.7.	TIMSELPCTL - Time range percentile values	77
2.8.8.	RUNSTAT - Running statistical values	78
2.8.9.	RUNPCTL - Running percentile values	79
2.8.10.	TIMSTAT - Statistical values over all time steps	80
2.8.11.	TIMPCTL - Percentile values over all time steps	81
2.8.12.	HOURLSTAT - Hourly statistical values	82
2.8.13.	HOURLPCTL - Hourly percentile values	83
2.8.14.	DAYSTAT - Daily statistical values	84
2.8.15.	DAYPCTL - Daily percentile values	85
2.8.16.	MONSTAT - Monthly statistical values	86
2.8.17.	MONPCTL - Monthly percentile values	87
2.8.18.	YEARSTAT - Yearly statistical values	88
2.8.19.	YEARPCTL - Yearly percentile values	89
2.8.20.	SEASSTAT - Seasonal statistical values	90
2.8.21.	SEASPCTL - Seasonal percentile values	91
2.8.22.	YHOURLSTAT - Multi-year hourly statistical values	92
2.8.23.	YDAYSTAT - Multi-year daily statistical values	93
2.8.24.	YDAYPCTL - Multi-year daily percentile values	94
2.8.25.	YMONSTAT - Multi-year monthly statistical values	95
2.8.26.	YMONPCTL - Multi-year monthly percentile values	96
2.8.27.	YSEASSTAT - Multi-year seasonal statistical values	97
2.8.28.	YSEASPCTL - Multi-year seasonal percentile values	98
2.8.29.	YDRUNSTAT - Multi-year daily running statistical values	99
2.8.30.	YDRUNPCTL - Multi-year daily running percentile values	101
2.9.	Regression	102
2.9.1.	REGRES - Regression	103
2.9.2.	DETREND - Detrend time series	103
2.9.3.	TREND - Trend of time series	104
2.9.4.	SUBTREND - Subtract a trend	104

2.10. Interpolation	105
2.10.1. REMAPGRID - SCRIP grid interpolation	106
2.10.2. GENWEIGHTS - Generate SCRIP grid interpolation weights	107
2.10.3. REMAP - SCRIP grid remapping	108
2.10.4. INTGRID - Grid interpolation	109
2.10.5. REMAPETA - Remap vertical hybrid level	110
2.10.6. INTVERT - Vertical interpolation	111
2.10.7. INTTIME - Time interpolation	112
2.10.8. INTYEAR - Year interpolation	113
2.11. Transformation	114
2.11.1. SPECTRAL - Spectral transformation	115
2.11.2. WIND - Wind transformation	116
2.12. Formatted I/O	117
2.12.1. INPUT - Formatted input	118
2.12.2. OUTPUT - Formatted output	119
2.13. Miscellaneous	120
2.13.1. GRIDCELL - Grid cell quantities	121
2.13.2. GRADSDS - GrADS data descriptor file	122
2.13.3. SMOOTH9 - 9 point smoothing	123
2.13.4. SETRANGE - Set range to constant	123
2.13.5. TIMSORT - Timsort	124
2.13.6. VARGEN - Generate a field	124
2.13.7. ROTUV - Rotation	125
2.13.8. MASTRFU - Mass stream function	125
2.13.9. HISTOGRAM - Histogram	126
2.13.10.WCT - Windchill temperature (C)	127
2.13.11.FDNS - Frost days where no snow index per time period	127
2.13.12.STRWIN - Strong wind days index per time period	127
2.13.13.STRBRE - Strong breeze days index per time period	128
2.13.14.STRGAL - Strong gale days index per time period	128
2.13.15.HURR - Hurricane days index per time period	128
2.14. Climate indices	129
2.14.1. ECACDD - Consecutive dry days index per time period	131
2.14.2. ECACFD - Consecutive frost days index per time period	131
2.14.3. ECACSU - Consecutive summer days index per time period	132
2.14.4. ECACWD - Consecutive wet days index per time period	132
2.14.5. ECACWDI - Cold wave duration index wrt mean of reference period	133
2.14.6. ECACWFI - Cold-spell days index wrt 10th percentile of reference period	133
2.14.7. ECAETR - Intra-period extreme temperature range	134
2.14.8. ECAFD - Frost days index per time period	134
2.14.9. ECAGSL - Growing season length index	135
2.14.10.ECAHD - Heating degree days per time period	136
2.14.11.ECAHWDI - Heat wave duration index wrt mean of reference period	136
2.14.12.ECAHWFI - Warm spell days index wrt 90th percentile of reference period	137
2.14.13.ECAID - Ice days index per time period	137
2.14.14.ECAR10MM - Heavy precipitation days index per time period	138
2.14.15.ECAR20MM - Very heavy precipitation days index per time period	138
2.14.16.ECAR75P - Moderate wet days wrt 75th percentile of reference period	139
2.14.17.ECAR75PTOT - Precipitation percent due to R75p days	139
2.14.18.ECAR90P - Wet days wrt 90th percentile of reference period	140
2.14.19.ECAR90PTOT - Precipitation percent due to R90p days	140
2.14.20.ECAR95P - Very wet days wrt 95th percentile of reference period	141
2.14.21.ECAR95PTOT - Precipitation percent due to R95p days	141
2.14.22.ECAR99P - Extremely wet days wrt 99th percentile of reference period	142
2.14.23.ECAR99PTOT - Precipitation percent due to R99p days	142
2.14.24.ECARR1 - Wet days index per time period	143
2.14.25.ECARX1DAY - Highest one day precipitation amount per time period	143
2.14.26.ECARX5DAY - Highest five-day precipitation amount per time period	144

2.14.27.ECASDII - Simple daily intensity index per time period . . . . .	144
2.14.28.ECASU - Summer days index per time period . . . . .	144
2.14.29.ECATG10P - Cold days percent wrt 10th percentile of reference period . . . . .	146
2.14.30.ECATG90P - Warm days percent wrt 90th percentile of reference period . . . . .	146
2.14.31.ECATN10P - Cold nights percent wrt 10th percentile of reference period . . . . .	147
2.14.32.ECATN90P - Warm nights percent wrt 90th percentile of reference period . . . . .	147
2.14.33.ECATR - Tropical nights index per time period . . . . .	148
2.14.34.ECATX10P - Very cold days percent wrt 10th percentile of reference period . . . . .	148
2.14.35.ECATX90P - Very warm days percent wrt 90th percentile of reference period . . . . .	149
<b>A. Hints for PINGO user</b>	<b>151</b>
<b>B. Grid description examples</b>	<b>152</b>
B.1. Example of a curvilinear grid description . . . . .	152
B.2. Example description for unstructured grid cells . . . . .	153
<b>Operator index</b>	<b>154</b>

# 1. Introduction

The Climate Data Operators (**CDO**) software is a collection of many operators for standard processing of climate and forecast model output. The operators include simple statistical and arithmetic functions, data selection and subsampling tools, and spatial interpolation. **CDO** was developed to have the same set of processing functions for [GRIB](#) and [netCDF](#) datasets in one package.

The Climate Data Interface (**CDI**) is used for the fast and file format independent access to GRIB and netCDF datasets. The local data formats SERVICE, EXTRA and IEG are also supported.

There are some limitations for GRIB and netCDF datasets. A GRIB dataset must be consistent, similar to netCDF. That means all time steps must have the same variables, and within a time step each variable may occur only once. NetCDF datasets are supported only with 2-dimensional, 3-dimensional and 4-dimensional variables and the attributes should follow the [GDT](#), [COARDS](#) or [CF Conventions](#).

The user interface and some operators are similar to the [PINGO](#) package. There are also some operators with the same name as in PINGO but with a different meaning. [Appendix A](#) gives an overview of those operators.

The main **CDO** features are:

- More than 400 operators available
- Modular design and easily extendable with new operators
- Very simple UNIX command line interface
- A dataset can be processed by several operators, without storing the interim results in files
- Most operators handle datasets with missing values
- Fast processing of large datasets
- Support of many different grid types
- Tested on many UNIX/Linux systems, Cygwin, and MacOS-X

## 1.1. Building from sources

This section describes how to build **CDO** from the sources on a UNIX system. **CDO** uses the GNU configure and build system to compile the source code. The only requirement is a working ANSI C compiler.

First go to the [download](#) page (<http://www.mpimet.mpg.de/cdo>) to get the latest distribution, if you do not already have it.

To take full advantage of **CDO** features the following additional library should be installed.

- Unidata [netCDF](#) library (<http://www.unidata.ucar.edu/packages/netcdf/index.html>) version 3 or higher. This is needed to read/write netCDF files with **CDO**.

### 1.1.1. Compilation

Compilation is now done by performing the following steps:

1. Unpack the archive, if you haven't already done that:

```
gunzip cdo-$VERSION.tar.gz    # uncompress the archive
tar xf cdo-$VERSION.tar       # unpack it
cd cdo-$VERSION
```

2. Run the configure script:

```
./configure
```

Or with netCDF support:

```
./configure --with-netcdf=<netCDF root directory>
```

For an overview of other configuration options use

```
./configure --help
```

3. Compile the program by running make:

```
make
```

The program should compile without problems and the binary (**cdo**) should be available in the **src** directory of the distribution.

### 1.1.2. Installation

After the compilation of the source code do a **make install**, possibly as root if the destination permissions require that.

```
make install
```

The binary is installed into the directory **<prefix>/bin**. **<prefix>** defaults to **/usr/local** but can be changed with the **--prefix** option of the configure script.

Alternatively, you can also copy the binary from the **src** directory manually to some **bin** directory in your search path.

## 1.2. Usage

This section describes how to use **CDO**. The syntax is:

```
cdo [Options] [Operators]
```

### 1.2.1. Options

All options must be placed before the first operator. The following options are available for all operators:

- a Convert from a relative to an absolute time axis.  
 -b <nbits> Set the number of bits for the output precision. The valid precisions depends on the file format:

<format>	<nbits>
grb	1 - 32
nc, nc2, srv, ext, ieg	32/64

For **srv**, **ext** and **ieg** format a **L** or **B** can be added to set the byteorder to Little or Big endian.

- f <format> Set the output file format. The valid file formats are:

File format	<format>
GRIB version 1	grb
netCDF	nc
netCDF version 2	nc2
SERVICE	srv
EXTRA	ext
IEG	ieg

- g <grid> Define the default grid description by name or from file.  
 Available grid names are: **t<RES>grid**, **r<NX>x<NY>**, **gme<NI>**  
 -h Help information for the operators.  
 -m <missval> Set the default missing value (default: **-9e+33**).  
 -R Convert GRIB data from reduced to regular grid.  
 -r Convert from an absolute to a relative time axis.  
 -s Silent mode.  
 -t <partab> Set the default parameter table name or file.  
 Predefined tables are: **echam4 echam5 mpiom1**  
 -V Print the version number.  
 -v Print extra details for some operators.  
 -z *gzip* Compress GRIB records with *gzip*.

### 1.2.2. Operators

There are more than 400 operators available. A detailed description of all operators can be found in the [Reference Manual](#) section.

### 1.2.3. Combining operators

All operators with a fixed number of input streams and one output stream can pipe the result directly to an other operator. The operator must begin with "-", in order to combine it with others. This can improve the performance by:

- reducing unnecessary disk I/O
- parallel processing

Use

```
cdo sub -dayavg ifile2 -timavg ifile1 ofile
```

instead of



```
cdo timavg ifile1 tmp1
cdo dayavg ifile2 tmp2
cdo sub tmp2 tmp1 ofile
rm tmp1 tmp2
```

### 1.2.4. Operator parameter

Some operators need one or more parameter.

- **STRING**

Unquoted characters without blanks and tabs. The following command select variables with the names `pressure` and `tsurf`:

```
cdo selvar,pressure,tsurf ifile ofile
```

- **FLOAT**

Floating point number in any representation. The following command sets the range between 0 and 273.15 of all fields to missing value:

```
cdo setrtomiss,0,273.15 ifile ofile
```

- **INTEGER**

A list of integers can be specified by *first/last[/inc]*. To select the days 5, 6, 7, 8 and 9 use:

```
cdo selday,5/9 ifile ofile
```

This is the same as:

```
cdo selday,5,6,7,8,9 ifile ofile
```

## 1.3. Grid description

In the following situations it is necessary to give a description of a horizontal grid:

- Changing the grid description (operator: `setgrid`)
- Horizontal interpolation (operator: `interpolate`, `remapXXX` and `genXXX`)
- Generating variables (operator: `const`, `random`)

As now described, there are several possibilities to define a horizontal grid. Predefined grids are available for global regular, gaussian or icosahedral-hexagonal GME grids.

### 1.3.1. Predefined grids

The following pre-defined grid names are available: `r<NX>x<NY>`, `t<RES>grid` and `gme<NI>`

#### Global regular grid: `r<NX>x<NY>`

`r<NX>x<NY>` defines a global regular grid. The number of the longitudes `<NX>` and the latitudes `<NY>` can be selected at will. The longitudes starts at 0° with an increment of  $(360/\text{<NX>})^\circ$ . The latitudes go from south to north with an increment of  $(180/\text{<NY>})^\circ$ .

## Global gaussian grid: t<RES>grid

t<RES>grid defines a global gaussian grid. Each valid triangular resolution can be used for <RES>. The longitudes starts at 0° with an increment of  $(360/nlon)^\circ$ . The gaussian latitudes go from north to south.

## Global icosahedral-hexagonal GME grid: gme<NI>

gme<NI> defines a global icosahedral-hexagonal GME grid. NI is the number of intervals on a main triangle side.

### 1.3.2. Grids from data files

You can use the grid description from an other datafile. The format of the datafile and the grid of the data field must be supported by this program. Use the operator 'sinfo' to get short informations about your variables and the grids. If there are more then one grid in the datafile the grid description of the first variable will be used.

### 1.3.3. SCRIP grids

SCRIP is a Spherical Coordinate Remapping and Interpolation Package. It is using a common grid description in netCDF. You can use it to describe curvilinear grids or unstructured grid cells. For more information about this format see [SCRIP]. This grid description format is only available if the program was compiled with netCDF support.

SCRIP grid description example of a curvilinear MPIOM1 GROB3 grid (only the netCDF header):

```
netcdf grob3s {
  dimensions:
    grid_size = 12120 ;
    grid_xsize = 120 ;
    grid_ysize = 101 ;
    grid_corners = 4 ;
    grid_rank = 2 ;
  variables:
    int grid_dims(grid_rank) ;
    float grid_center_lat(grid_ysize, grid_xsize) ;
      grid_center_lat:units = "degrees" ;
      grid_center_lat:bounds = "grid_corner_lat" ;
    float grid_center_lon(grid_ysize, grid_xsize) ;
      grid_center_lon:units = "degrees" ;
      grid_center_lon:bounds = "grid_corner_lon" ;
    int grid_imask(grid_ysize, grid_xsize) ;
      grid_imask:units = "unitless" ;
      grid_imask:coordinates = "grid_center_lon grid_center_lat" ;
    float grid_corner_lat(grid_ysize, grid_xsize, grid_corners) ;
      grid_corner_lat:units = "degrees" ;
    float grid_corner_lon(grid_ysize, grid_xsize, grid_corners) ;
      grid_corner_lon:units = "degrees" ;

    // global attributes:
      :title = "grob3s" ;
}
```

### 1.3.4. PINGO grids

PINGO uses a very simple grid description in ASCII format to describe regular longitude/latitude or global gaussian grids. All PINGO grid description files are supported by CDO. For more information about this format see [PINGO].

PINGO grid description example of a T21 gaussian grid:

```

Grid Description File
(Comments start at non digit characters and end at end of line)
First part: The dimensions.
64 32 = Number of longitudes and latitudes
Second part: The listed longitudes.
2 means equidistant longitudes
0.000000 5.625000 = Most western and second most western longitude
Third part: The listed latitudes.
32 means all 32 latitudes are given in the following list:
85.761 80.269 74.745 69.213 63.679 58.143 52.607 47.070
41.532 35.995 30.458 24.920 19.382 13.844 8.307 2.769
-2.769 -8.307 -13.844 -19.382 -24.920 -30.458 -35.995 -41.532
-47.070 -52.607 -58.143 -63.679 -69.213 -74.745 -80.269 -85.761

```

### 1.3.5. CDO grids

All supported grids can be also described with the **CDO** description ASCII formatted file. The following keywords can be used to describe a grid:

Keyword	Datatype	Description
<b>gridtype</b>	STRING	type of the grid (gaussian, lonlat, curvilinear, cell)
<b>gridsize</b>	INTEGER	size of the grid
<b>xsize</b>	INTEGER	size in x direction (number of longitudes)
<b>ysize</b>	INTEGER	size in y direction (number of latitudes)
<b>xvals</b>	FLOAT ARRAY	x values of the grid
<b>yvals</b>	FLOAT ARRAY	y values of the grid
<b>xnpole</b>	FLOAT	x value of the north pole (rotated grid)
<b>ynpole</b>	FLOAT	y value of the north pole (rotated grid)
<b>nvertex</b>	INTEGER	number of the vertices for all grid cells
<b>xbounds</b>	FLOAT ARRAY	x bounds of each gridbox
<b>ybounds</b>	FLOAT ARRAY	y bounds of each gridbox
<b>xfirst, xinc</b>	FLOAT, FLOAT	macros to define xvals with a constant increment
<b>yfirst, yinc</b>	FLOAT, FLOAT	macros to define yvals with a constant increment

Which keywords are necessary depends on the gridtype. The following table gives an overview of the default values or the size for the different grid types.

gridtype	lonlat	gaussian	curvilinear	cell
gridsize	xsize*ysize	xsize*ysize	xsize*ysize	<b>ncell</b>
xsize	<b>nlon</b>	<b>nlon</b>	<b>nlon</b>	gridsize
ysize	<b>nlat</b>	<b>nlat</b>	<b>nlat</b>	gridsize
xvals	xsize	xsize	gridsize	gridsize
yvals	ysize	ysize	gridsize	gridsize
xnpole	0			
ynpole	90			
nvertex	2	2	4	<b>nv</b>
xbounds	2*xsize	2*xsize	4*gridsize	nv*gridsize
ybounds	2*ysize	2*ysize	4*gridsize	nv*gridsize

The keywords nvertex, xbounds and ybounds are optional if the area weights are not needed.

**CDO** grid description example of a T21 gaussian grid:

```

gridtype = gaussian
xsize    = 64
ysize    = 32

```

```

xfirst = 0
xinc   = 5.625
yvals  = 85.76  80.27  74.75  69.21  63.68  58.14  52.61  47.07
        41.53  36.00  30.46  24.92  19.38  13.84  8.31  2.77
        -2.77 -8.31 -13.84 -19.38 -24.92 -30.46 -36.00 -41.53
        -47.07 -52.61 -58.14 -63.68 -69.21 -74.75 -80.27 -85.76

```

**CDO** grid description example of a global regular grid with 60x30 points:

```

gridtype = lonlat
xsize    = 60
ysize    = 30
xfirst   = -177
xinc     = 6
yfirst   = -87
yinc     = 6

```

For a lon/lat grid with a rotated pole, the north pole must be defined. As far as you define the keywords `xnpole`/`ynpole` all coordinate values are for the rotated system.

**CDO** grid description example of a regional rotated lon/lat grid:

```

gridtype = lonlat
xsize    = 81
ysize    = 91
xfirst   = -19.5
xinc     = 0.5
yfirst   = -25.0
yinc     = 0.5
xnpole   = -170
ynpole   = 32.5

```

Example **CDO** descriptions of a curvilinear and an unstructured grid can be found in [Appendix B](#).

## 1.4. Z-axis description

Sometimes it is necessary to change the description of a z-axis. This can be done with the operator [setzaxis](#). This operator needs an ASCII formatted file with the description of the z-axis. The following keywords can be used to describe a z-axis:

Keyword	Datatype	Description
<b>zaxistype</b>	STRING	type of the z-axis
<b>size</b>	INTEGER	number of levels
<b>levels</b>	FLOAT ARRAY	values of the levels
<b>lbounds</b>	FLOAT ARRAY	lower level bounds
<b>ubounds</b>	FLOAT ARRAY	upper level bounds
<b>vctsize</b>	INTEGER	number of vertical coordinate parameters
<b>vct</b>	FLOAT ARRAY	vertical coordinate table

The keywords **lbounds** and **ubounds** are optional. **vctsize** and **vct** are only necessary to define hybrid model levels.

Available z-axis types:

Z-axis type	Description	Units
<b>surface</b>	Surface	
<b>pressure</b>	Pressure level	pascal
<b>hybrid</b>	Hybrid model level	
<b>height</b>	Height above ground	meter
<b>depth_below_sea</b>	Depth below sea level	meter
<b>depth_below_land</b>	Depth below land surface	centimeter
<b>isentropic</b>	Isentropic (theta) level	kelvin

Z-axis description example for pressure levels 100, 200, 500, 850 and 1000 hPa:

```
zaxistype = pressure
size      = 5
levels    = 10000 20000 50000 85000 100000
```

## 1.5. Time axis

A time axis describes the time for every timestep. Two time axis types are available: absolute time and relative time axis. **CDO** tries to maintain the actual type of the time axis for all operators. The operators for time range statistic (e.g.: monavg, ymonavg, ...) create an absolute time axis.

### 1.5.1. Absolute time

An absolute time axis has the current time to each time step. It can be used without knowledge of the calendar. This is preferably used by climate models. In netCDF files the relative time axis is represented by the unit of the time: "day as %Y%m%d.%f".

### 1.5.2. Relative time

A relative time is the time relative to a fixed reference time. The current time results from the reference time and the elapsed interval. The result depends on the calendar used. **CDO** supports the standard Gregorian, 360 days, 365 days and 366 days calendars. The relative time axis is preferably used by weather forecast models. In netCDF files the relative time axis is represented by the unit of the time: "time-units since reference-time", e.g "days since 1989-6-15 12:00".

### 1.5.3. Conversion of the time

Some programs which work with netCDF data can only process relative time axes. Therefore it may be necessary to convert from an absolute into a relative time axis. This conversion can be done for each operator with the **CDO** option '-r'. To convert a relative into an absolute time axis use the **CDO** option '-a'.

## 1.6. Parameter table

A parameter table is an ASCII formatted file to convert code numbers to variable names. Each variable has one line with the code number, the name and the description with optional units in a blank separated list. It can be used only for GRIB, SERVICE, EXTRA and IEG formatted files. The **CDO** option '-t <partab>' sets the default parameter table for all input files. Use the operator 'setpartab' to set the parameter table for a specific file.

Example of a **CDO** parameter table:

134	aps	surface pressure [Pa]
141	sn	snow depth [m]
147	ahfl	latent heat flux [W/m**2]
172	slm	land sea mask
175	albedo	surface albedo
211	siced	ice depth [m]

## 1.7. Missing values

Most operators can handle missing values. The default missing value for GRIB, SERVICE, EXTRA and IEG files is  $-9e + 33$ . The **CDO** option '`-m <missval>`' overwrites the default missing value. In netCDF files the variable attribute '`_FillValue`' is used as a missing value. The operator '[setmissval](#)' can be used to set a new missing value.

The **CDO** use of the missing value is shown in the following tables, where one table is printed for each operation. The operations are applied to arbitrary numbers  $a$ ,  $b$ , the special case 0, and the missing value *miss*. For example the table named "addition" shows that the sum of an arbitrary number  $a$  and the missing value is the missing value, and the table named "multiplication" shows that 0 multiplied by missing value results in 0.

addition	b	miss	
a	$a + b$	miss	
miss	miss	miss	
subtraction	b	miss	
a	$a - b$	miss	
miss	miss	miss	
multiplication	b	0	miss
a	$a * b$	0	miss
0	0	0	0
miss	miss	0	miss
division	b	0	miss
a	$a/b$	miss	miss
0	0	miss	miss
miss	miss	miss	miss
maximum	b	miss	
a	$\max(a, b)$	a	
miss	b	miss	
minimum	b	miss	
a	$\min(a, b)$	a	
miss	b	miss	

The handling of missing values by the operations "minimum" and "maximum" may be surprising, but the definition given here is more consistent with that expected in practice. Mathematical functions (e.g. *log*, *sqrt*, etc.) return the missing value if an argument is the missing value or an argument is out of range.

All statistical functions ignore missing values, treating them as not belonging to the sample, with the side-effect of a reduced sample size.

### 1.7.1. Mean and average

An artificial distinction is made between the notions mean and average. The mean is regarded as a statistical function, whereas the average is found simply by adding the sample members and dividing the result by the sample size. For example, the mean of 1, 2, *miss* and 3 is  $(1 + 2 + 3)/3 = 2$ , whereas the average is  $(1 + 2 + \textit{miss} + 3)/4 = \textit{miss}/4 = \textit{miss}$ . If there are no missing values in the sample, the average and mean are identical.

## 2. Reference manual

This section gives a description of all operators. Similar operators are grouped to modules. For easier description all single input files are named `ifile` or `ifile1`, `ifile2`, etc., and an unlimited number of input files are named `ifiles`. All output files are named `ofile` or `ofile1`, `ofile2`, etc. Further the following notion is introduced:

$i(t)$	Timestep $t$ of <code>ifile</code>
$i(t, x)$	Element number $x$ of the field at timestep $t$ of <code>ifile</code>
$o(t)$	Timestep $t$ of <code>ofile</code>
$o(t, x)$	Element number $x$ of the field at timestep $t$ of <code>ofile</code>



## 2.1. Information

This section contains modules to print information about datasets. All operators print their results to standard output.

Here is a short overview of all operators in this section:

<b>info</b>	Dataset information listed by code number
<b>infov</b>	Dataset information listed by variable name
<b>map</b>	Dataset information and simple map
<b>sinfo</b>	Short dataset information listed by code number
<b>sinfov</b>	Short dataset information listed by variable name
<b>diff</b>	Compare two datasets listed by code number
<b>diffv</b>	Compare two datasets listed by variable name
<b>npar</b>	Number of parameters
<b>nlevel</b>	Number of levels
<b>nyear</b>	Number of years
<b>nmon</b>	Number of months
<b>ndate</b>	Number of dates
<b>ntime</b>	Number of time steps
<b>showformat</b>	Show file format
<b>showcode</b>	Show code numbers
<b>showname</b>	Show variable names
<b>showstdname</b>	Show standard names
<b>showlevel</b>	Show levels
<b>showltype</b>	Show GRIB level types
<b>showyear</b>	Show years
<b>showmon</b>	Show months
<b>showdate</b>	Show dates
<b>showtime</b>	Show time steps
<b>pardes</b>	Parameter description
<b>griddes</b>	Grid description
<b>zaxisdes</b>	Zaxis description
<b>vct</b>	Vertical coordinate table

### 2.1.1. INFO - Information and simple statistics

#### Synopsis

```
<operator> ifiles
```

#### Description

This module writes information about the structure and contents of all input datasets to standard output. The information displayed depends on the actual operator.

#### Operators

- info** Dataset information listed by code number  
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
  - Code number and Level
  - Size of the grid and number of Missing values
  - Minimum, Mean and Maximum
- The mean value is computed without the use of area weights!
- infov** Dataset information listed by variable name  
The same as operator [info](#) but using the name instead of the code number to identify the variables.
- 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 rectangular grid.

#### Example

To print information and simple statistics for each field of a dataset use:

```
cdo info ifile
```

This is an example result of a dataset with one 2D variable over 12 time steps:

-1 :	Date	Time	Code	Level	Size	Miss :	Minimum	Mean	Maximum
1 :	1987-01-31	12:00	139	0	2048	1361 :	232.77	266.65	305.31
2 :	1987-02-28	12:00	139	0	2048	1361 :	233.64	267.11	307.15
3 :	1987-03-31	12:00	139	0	2048	1361 :	225.31	267.52	307.67
4 :	1987-04-30	12:00	139	0	2048	1361 :	215.68	268.65	310.47
5 :	1987-05-31	12:00	139	0	2048	1361 :	215.78	271.53	312.49
6 :	1987-06-30	12:00	139	0	2048	1361 :	212.89	272.80	314.18
7 :	1987-07-31	12:00	139	0	2048	1361 :	209.52	274.29	316.34
8 :	1987-08-31	12:00	139	0	2048	1361 :	210.48	274.41	315.83
9 :	1987-09-30	12:00	139	0	2048	1361 :	210.48	272.37	312.86
10 :	1987-10-31	12:00	139	0	2048	1361 :	219.46	270.53	309.51
11 :	1987-11-30	12:00	139	0	2048	1361 :	230.98	269.85	308.61
12 :	1987-12-31	12:00	139	0	2048	1361 :	241.25	269.94	309.27

## 2.1.2. SINFO - Short information

### Synopsis

```
<operator> ifiles
```

### Description

This module writes information about the structure of all input datasets to standard output. The information displayed depends on the actual operator.

### Operators

- sinfo** Short dataset information listed by code number  
Prints short information of a dataset. The information is divided into 4 sections. Section 1 prints one line per variable with the following information:
- institute and source
  - parameter table and code number
  - horizontal grid size and number
  - number of vertical levels and zaxis number
- Section 2 and 3 gives a short overview of all horizontal and vertical grids. And the last section contains short information of the time axis.
- sinfov** Short dataset information listed by variable name  
The same as operator [sinfo](#) but using the name instead of the code number and parameter table to identify the variables.

### Example

To print short information of a dataset use:

```
cdo sinfo ifile
```

This is the result of an ECHAM5 dataset with 3 variables over 12 time steps:

```
-1 : Institut Source Table Code Time Typ Grid Size Num Levels Num
 1 : MPIMET ECHAM5.3 128 129 constant F32 2048 1 1 1
 2 : MPIMET ECHAM5.3 128 130 variable F32 2048 1 4 2
 3 : MPIMET ECHAM5.3 128 139 variable F32 2048 1 1 1
Horizontal grids :
 1 : gaussian > size : dim = 2048 nlon = 64 nlat = 32
      longitude : first = 0 last = 354.375 inc = 5.625
      latitude : first = 85.7605871 last = -85.7605871
Vertical grids :
 1 : surface : 0
 2 : pressure Pa : 92500 85000 50000 20000
Time axis : 12 steps
YYYY-MM-DD hh:mm YYYY-MM-DD hh:mm YYYY-MM-DD hh:mm YYYY-MM-DD hh:mm
1987-01-31 12:00 1987-02-28 12:00 1987-03-31 12:00 1987-04-30 12:00
1987-05-31 12:00 1987-06-30 12:00 1987-07-31 12:00 1987-08-31 12:00
1987-09-30 12:00 1987-10-31 12:00 1987-11-30 12:00 1987-12-31 12:00
```

### 2.1.3. DIFF - Compare two datasets field by field

#### Synopsis

```
<operator> ifile1 ifile2
```

#### Description

Compares the contents of two datasets field by field. The input datasets must have the same structure and the fields must have the same header information and dimensions.

#### Operators

**diff** Compare two datasets listed by code number  
Provides statistics on differences between two datasets. For each pair of fields the operator prints one line with the following information:

- date and time
- code number and level
- size of the grid and number of missing values
- occurrence of coefficient pairs with different signs
- occurrence of zero values
- maxima of absolute difference of coefficient pairs
- maxima of relative difference of non-zero coefficient pairs with equal signs

**diffv** Compare two datasets listed by variable name  
The same as operator [diff](#). Using the name instead of the code number to identify the variable.

#### Example

To print the difference for each field of two datasets use:

```
cdo diff ifile1 ifile2
```

This is an example result of the difference of two datasets with one 2D variable over 12 time steps:

	Date	Time	Code	Level	Size	Miss	: S Z	Absdiff	Reldiff
1	: 1987-01-31	12:00	139	0	2048	1361	: F F	0.00010681	4.1660e-07
2	: 1987-02-28	12:00	139	0	2048	1361	: F F	6.1035e-05	2.3742e-07
3	: 1987-03-31	12:00	139	0	2048	1361	: F F	7.6294e-05	3.3784e-07
4	: 1987-04-30	12:00	139	0	2048	1361	: F F	7.6294e-05	3.5117e-07
5	: 1987-05-31	12:00	139	0	2048	1361	: F F	0.00010681	4.0307e-07
6	: 1987-06-30	12:00	139	0	2048	1361	: F F	0.00010681	4.2670e-07
7	: 1987-07-31	12:00	139	0	2048	1361	: F F	9.1553e-05	3.5634e-07
8	: 1987-08-31	12:00	139	0	2048	1361	: F F	7.6294e-05	2.8849e-07
9	: 1987-09-30	12:00	139	0	2048	1361	: F F	7.6294e-05	3.6168e-07
10	: 1987-10-31	12:00	139	0	2048	1361	: F F	9.1553e-05	3.5001e-07
11	: 1987-11-30	12:00	139	0	2048	1361	: F F	6.1035e-05	2.3839e-07
12	: 1987-12-31	12:00	139	0	2048	1361	: F F	9.3553e-05	3.7624e-07

## 2.1.4. NINFO - Print the number of parameters, levels or times

### Synopsis

`<operator> ifile`

### Description

This module prints, according to the actual operator, the number of variables, levels or times of the input dataset.

### Operators

<b>npar</b>	Number of parameters Prints the number of parameters (variables).
<b>nlevel</b>	Number of levels Prints the number of levels for each variable.
<b>nyear</b>	Number of years Prints the number of different years.
<b>nmon</b>	Number of months Prints the number of different combinations of years and months.
<b>ndate</b>	Number of dates Prints the number of different dates.
<b>ntime</b>	Number of time steps Prints the number of time steps.

### Example

To print the number of parameters (variables) in a dataset use:

```
cdo npar ifile
```

To print the number of month in a dataset use:

```
cdo nmon ifile
```

## 2.1.5. SHOWINFO - Show variables, levels or times

### Synopsis

```
<operator> ifile
```

### Description

This module prints, according to the actual operator, the format, variables, levels or times of the input dataset.

### Operators

<b>showformat</b>	Show file format Prints the file format of the input dataset.
<b>showcode</b>	Show code numbers Prints the code number of all different variables.
<b>showname</b>	Show variable names Prints the name of all different variables.
<b>showstdname</b>	Show standard names Prints the standard name of all different variables.
<b>showlevel</b>	Show levels Prints all levels for each variable.
<b>showltype</b>	Show GRIB level types Prints the GRIB level type for all different z-axis.
<b>showyear</b>	Show years Prints all different years.
<b>showmon</b>	Show months Prints all different months.
<b>showdate</b>	Show dates Prints all different dates.
<b>showtime</b>	Show time steps Prints all time steps.

### Example

To print the code number of all variables in a dataset use:

```
cdo showcode ifile
```

This is an example result of a dataset with three variables:

```
129 130 139
```

To print all months in a dataset use:

```
cdo showmon ifile
```

This is an examples result of a dataset with an annual cycle:

```
1 2 3 4 5 6 7 8 9 10 11 12
```

## 2.1.6. FILEDES - Dataset description

### Synopsis

```
<operator> ifile
```

### Description

This module prints, according to the actual operator, the description of the parameters, the grids, the z-axis or the vertical coordinate table.

### Operators

<b>pardes</b>	Parameter description Prints a table with a description of all variables. For each variable the operator prints one line listing the code, name, description and units.
<b>griddes</b>	Grid description Prints the description of all grids.
<b>zaxisdes</b>	Zaxis description Prints the description of all z-axis.
<b>vct</b>	Vertical coordinate table Prints the vertical coordinate table.

### Example

Assume all variables of the dataset are on a T21 gaussssian grid. To print the grid description of this dataset use:

```
cdo griddes ifile
```

Result:

```
gridtype : gaussian
gridsize : 2048
xname    : lon
xlongname : longitude
xunits   : degrees_east
yname    : lat
ylongname : latitude
yunits   : degrees_north
xsize    : 64
ysize    : 32
xfirst   : 0
xinc     : 5.625
yvals    : 85.76058 80.26877 74.74454 69.21297 63.67863 58.1429 52.6065
           47.06964 41.53246 35.99507 30.4575 24.91992 19.38223 13.84448
           8.306702 2.768903 -2.768903 -8.306702 -13.84448 -19.38223
           -24.91992 -30.4575 -35.99507 -41.53246 -47.06964 -52.6065
           -58.1429 -63.67863 -69.21297 -74.74454 -80.26877 -85.76058
```

## 2.2. File operations

This section contains modules to perform operations on files.

Here is a short overview of all operators in this section:

<b>copy</b>	Copy datasets
<b>cat</b>	Concatenate datasets
<b>replace</b>	Replace variables
<b>merge</b>	Merge datasets with different fields
<b>mergetime</b>	Merge datasets sorted by date and time
<b>splitcode</b>	Split code numbers
<b>splitname</b>	Split variable names
<b>splitlevel</b>	Split levels
<b>splitgrid</b>	Split grids
<b>splitzaxis</b>	Split zaxis
<b>splithour</b>	Split hours
<b>splitday</b>	Split days
<b>splitmon</b>	Split months
<b>splitseas</b>	Split seasons
<b>splityear</b>	Split years
<b>splitsel</b>	Split time selection



### 2.2.1. COPY - Copy datasets

#### Synopsis

```
<operator> ifiles ofile
```

#### Description

This module contains operators to copy or concatenate datasets. Each input dataset must have the same variables with complete time steps.

#### Operators

<b>copy</b>	Copy datasets Copies all input datasets to <b>ofile</b> .
<b>cat</b>	Concatenate datasets Concatenates all input datasets and append the result to the end of <b>ofile</b> . If <b>ofile</b> does not exist it will be created.

#### Example

To change the format of a dataset to netCDF use:

```
cdo -f nc copy ifile ofile.nc
```

Add the option '-r' to create a relative time axis, as is required for proper recognition by GrADS or Ferret:

```
cdo -r -f nc copy ifile ofile.nc
```

To concatenate 3 datasets with different time steps of the same variables use:

```
cdo copy ifile1 ifile2 ifile3 ofile
```

If the output dataset already exist and you wish to extend it with more time steps use:

```
cdo cat ifile1 ifile2 ifile3 ofile
```

### 2.2.2. REPLACE - Replace variables

#### Synopsis

```
replace ifile1 ifile2 ofile
```

#### Description

Replaces all common variables of **ifile2** and **ifile1** with those of **ifile1** and write the result to **ofile**. Both input datasets must have the same number of time steps.

#### Example

Assume the first input dataset **ifile1** has three variables with the names **geosp**, **t** and **tslm1** and the second input dataset **ifile2** has only the variable **tslm1**. To replace the variable **tslm1** in **ifile1** with **tslm1** from **ifile2** use:

```
cdo replace ifile1 ifile2 ofile
```

### 2.2.3. MERGE - Merge datasets

#### Synopsis

```
<operator> ifiles ofile
```

#### Description

This module reads datasets from several input files, merges them and writes the resulting dataset to `ofile`.

#### Operators

<b>merge</b>	Merge datasets with different fields Merges time series of different fields from several input datasets. The number of fields per time step written to <code>ofile</code> is the sum of the field numbers per time step in all input datasets. The time series on all input datasets must have different fields and the same number of time steps.
<b>mergetime</b>	Merge datasets sorted by date and time Merges all time steps of all input files sorted by date and time. After this operation every input time step is in <code>ofile</code> and all time steps are sorted by date and time. Each input file must have the same variables and different time steps.

#### Example

Assume three datasets with the same number of time steps and each dataset with different variables. To merge these datasets to a new dataset use:

```
cdo merge ifile1 ifile2 ifile3 ofile
```

Assume you have split a 6 hourly dataset with [splithour](#). This produces four datasets one for each hour. The following command merges them together:

```
cdo mergetime ifile1 ifile2 ifile3 ifile4 ofile
```

## 2.2.4. SPLIT - Split a dataset

### Synopsis

```
<operator> ifile oprefix
```

### Description

This module splits a dataset to several files with names formed from the field header information and **oprefix**.

### Operators

<b>splitcode</b>	Split code numbers Splits a dataset into pieces, one for each different code number. Appends three digits with the code number to <b>oprefix</b> to form the output file names.
<b>splitname</b>	Split variable names Splits a dataset into pieces, one for each variable name. Appends a string with the variable name to <b>oprefix</b> to form the output file names.
<b>splitlevel</b>	Split levels Splits a dataset into pieces, one for each different level. Appends six digits with the level to <b>oprefix</b> to form the output file names.
<b>splitgrid</b>	Split grids Splits a dataset into pieces, one for each different grid. Appends two digits with the grid number to <b>oprefix</b> to form the output file names.
<b>splitzaxis</b>	Split zaxis Splits a dataset into pieces, one for each different zaxis. Appends two digits with the zaxis number to <b>oprefix</b> to form the output file names.

### Example

Assume an input GRIB dataset with three variables, e.g. code number 129, 130 and 139. To split this dataset into three pieces, one for each code number use:

```
cdo splitcode ifile code
```

Result of 'dir code\*':

```
code129.grb code130.grb code139.grb
```

## 2.2.5. SPLITTIME - Split time steps of a dataset

### Synopsis

```
<operator> ifile oprefix
```

### Description

This module splits time steps of a dataset to several files with names formed from the field header information and **oprefix**.

### Operators

<b>splithour</b>	Split hours Splits a file into pieces, one for each different hour. Appends two digits with the hour to <b>oprefix</b> to form the output file names.
<b>splitday</b>	Split days Splits a file into pieces, one for each different day. Appends two digits with the day to <b>oprefix</b> to form the output file names.
<b>splitmon</b>	Split months Splits a file into pieces, one for each different month. Appends two digits with the month to <b>oprefix</b> to form the output file names.
<b>splitseas</b>	Split seasons Splits a file into pieces, one for each different season. Appends three characters with the season to <b>oprefix</b> to form the output file names.
<b>splityear</b>	Split years Splits a file into pieces, one for each different year. Appends four digits with the year to <b>oprefix</b> to form the output file names.

### Example

Assume the input GRIB dataset has time steps from January to December. To split each month with all variables into one separate file use:

```
cdo splitmon ifile mon
```

Result of 'dir mon\*':

```
mon01.grb  mon02.grb  mon03.grb  mon04.grb  mon05.grb  mon06.grb
mon07.grb  mon08.grb  mon09.grb  mon10.grb  mon11.grb  mon12.grb
```

## 2.2.6. SPLITSEL - Split selected time steps

### Synopsis

```
splitsel,nsets[,noffset[,nskip]] ifile oprefix
```

### Description

This operator splits a dataset into pieces, one for each adjacent sequence  $t_1, \dots, t_n$  of time steps of the same selected time range. Appends three digits with the sequence number to **oprefix** to form the output file names.

### Parameter

<i>nsets</i>	INTEGER	Number of input time steps for each output file
<i>noffset</i>	INTEGER	Number of input time steps skipped before the first time step range (optional)
<i>nskip</i>	INTEGER	Number of input time steps skipped between time step ranges (optional)

## 2.3. Selection

This section contains modules to select time steps, fields or part of a field from a dataset.

Here is a short overview of all operators in this section:

<b>selcode</b>	Select variables by code number
<b>delcode</b>	Delete variables by code number
<b>selname</b>	Select variables by name
<b>delname</b>	Delete variables by name
<b>selstdname</b>	Select variables by standard name
<b>sellevel</b>	Select levels
<b>selgrid</b>	Select grids
<b>selgridname</b>	Select grids by name
<b>selzaxis</b>	Select zaxes
<b>selzaxisname</b>	Select zaxes by name
<b>selltype</b>	Select GRIB level types
<b>seltabnum</b>	Select parameter table numbers
<b>sel timestep</b>	Select time steps
<b>seltime</b>	Select times
<b>selhour</b>	Select hours
<b>selday</b>	Select days
<b>selmon</b>	Select months
<b>selyear</b>	Select years
<b>sel seas</b>	Select seasons
<b>seldate</b>	Select dates
<b>selsmon</b>	Select single month
<b>sellonlatbox</b>	Select a longitude/latitude box
<b>selindexbox</b>	Select an index box

### 2.3.1. SELECT - Select fields

#### Synopsis

```

selcode, codes ifile ofile
delcode, codes ifile ofile
selname, varnames ifile ofile
delname, varnames ifile ofile
selstdname, stdnames ifile ofile
sellevel, levels ifile ofile
selgrid, grids ifile ofile
selgridname, gridnames ifile ofile
selzaxis, zaxes ifile ofile
selzaxisname, zaxisnames ifile ofile
selltype, ltypes ifile ofile
seltabnum, tabnums ifile ofile

```

#### Description

This module selects some fields from **ifile** and writes them to **ofile**. The fields selected depend on the actual operator and the parameters.

#### Operators

<b>selcode</b>	Select variables by code number Selects all fields with code numbers in a user given list.
<b>delcode</b>	Delete variables by code number Deletes all fields with code numbers in a user given list.
<b>selname</b>	Select variables by name Selects all fields with variable names in a user given list.
<b>delname</b>	Delete variables by name Deletes all fields with variable names in a user given list.
<b>selstdname</b>	Select variables by standard name Selects all fields with standard names in a user given list.
<b>sellevel</b>	Select levels Selects all fields with levels in a user given list.
<b>selgrid</b>	Select grids Selects all fields with grids in a user given list.
<b>selgridname</b>	Select grids by name Selects all fields with grid names in a user given list.
<b>selzaxis</b>	Select zaxes Selects all fields with zaxes in a user given list.
<b>selzaxisname</b>	Select zaxes by name Selects all fields with zaxis names in a user given list.
<b>selltype</b>	Select GRIB level types Selects all fields with GRIB level type in a user given list.
<b>seltabnum</b>	Select parameter table numbers Selects all fields with parameter table numbers in a user given list.

## Parameter

<i>codes</i>	INTEGER	Comma separated list of code numbers
<i>varnames</i>	STRING	Comma separated list of variable names
<i>stdnames</i>	STRING	Comma separated list of standard names
<i>levels</i>	FLOAT	Comma separated list of levels
<i>ltypes</i>	INTEGER	Comma separated list of GRIB level types
<i>grids</i>	INTEGER	Comma separated list of grid numbers
<i>gridnames</i>	STRING	Comma separated list of grid names
<i>zaxes</i>	INTEGER	Comma separated list of zaxis numbers
<i>zaxisnames</i>	STRING	Comma separated list of zaxis names
<i>tabnums</i>	INTEGER	Comma separated list of parameter table numbers

## Example

Assume an input dataset has three variables with the code numbers 129, 130 and 139. To select the variables with the code number 129 and 139 use:

```
cdo selcode,129,139 ifile ofile
```

You can also select the code number 129 and 139 by deleting the code number 130 with:

```
cdo delcode,130 ifile ofile
```



### 2.3.2. SELTIME - Select time steps

#### Synopsis

```

se timestep,timesteps ifile ofile

sel time,times ifile ofile

sel hour,hours ifile ofile

sel day,days ifile ofile

sel mon,months ifile ofile

sel year,years ifile ofile

sel seas,seasons ifile ofile

sel date,date1[,date2] ifile ofile

sel smon,month[,nts1[,nts2]] ifile ofile

```

#### Description

This module selects user specified time steps from **ifile** and writes them to **ofile**. The time steps selected depends on the actual operator and the parameters.

#### Operators

<b>se timestep</b>	Select time steps Selects all time steps with a time step in a user given list.
<b>sel time</b>	Select times Selects all time steps with a time in a user given list.
<b>sel hour</b>	Select hours Selects all time steps with a hour in a user given list.
<b>sel day</b>	Select days Selects all time steps with a day in a user given list.
<b>sel mon</b>	Select months Selects all time steps with a month in a user given list.
<b>sel year</b>	Select years Selects all time steps with a year in a user given list.
<b>sel seas</b>	Select seasons Selects all time steps with a month of a season in a user given list.
<b>sel date</b>	Select dates Selects all time steps with a date in a user given range.
<b>sel smon</b>	Select single month Selects a month and optional an unlimited number of time steps before and after this month.

**Parameter**

<i>timesteps</i>	INTEGER	Comma separated list of time steps
<i>times</i>	STRING	Comma separated list of times (format hh:mm)
<i>hours</i>	INTEGER	Comma separated list of hours
<i>days</i>	INTEGER	Comma separated list of days
<i>months</i>	INTEGER	Comma separated list of months
<i>years</i>	INTEGER	Comma separated list of years
<i>seasons</i>	STRING	Comma separated list of seasons (DJF, MAM, JJA, SON)
<i>date1</i>	STRING	Start date (format YYYY-MM-DDThh:mm)
<i>date2</i>	STRING	End date (format YYYY-MM-DDThh:mm)
<i>nts1</i>	INTEGER	Number of time steps before the selected month [default: 0]
<i>nts2</i>	INTEGER	Number of time steps after the selected month [default: nts1]

### 2.3.3. SELBOX - Select a box of a field

#### Synopsis

```
sellonlatbox,lon1,lon2,lat1,lat2 ifile ofile
```

```
selindexbox,idx1,idx2,idy1,idy2 ifile ofile
```

#### Description

Selects a box of the rectangular understood field. All input fields must have the same horizontal grid.

#### Operators

<b>sellonlatbox</b>	Select a longitude/latitude box Selects a longitude/latitude box. The user has to give the longitudes and latitudes of the edges of the box.
<b>selindexbox</b>	Select an index box Selects an index box. The user has to give the indexes of the edges of the box. The index of the left edge may be greater then that of the right edge.

#### Parameter

<i>lon1</i>	FLOAT	Western longitude
<i>lon2</i>	FLOAT	Eastern longitude
<i>lat1</i>	FLOAT	Southern or northern latitude
<i>lat2</i>	FLOAT	Northern or southern latitude
<i>idx1</i>	INTEGER	Index of first longitude
<i>idx2</i>	INTEGER	Index of last longitude
<i>idy1</i>	INTEGER	Index of first latitude
<i>idy2</i>	INTEGER	Index of last latitude

#### Example

To select the region with the longitudes from 120E to 90W and latitudes from 20N to 20S from all input fields use:

```
cdo sellonlatbox,120,-90,20,-20 ifile ofile
```

If the input dataset has fields on a T21 gaussian grid, the same box can be selected with [selindexbox](#) by:

```
cdo selindexbox,23,48,13,20 ifile ofile
```

## 2.4. Conditional selection

This section contains modules to conditional select field elements. The fields in the first input file are handled as a mask. A value not equal to zero is treated as "true", zero is treated as "false".

Here is a short overview of all operators in this section:

<b>ifthen</b>	If then
<b>ifnotthen</b>	If not then
<b>ifthenelse</b>	If then else
<b>ifthenc</b>	If then constant
<b>ifnotthenc</b>	If not then constant

### 2.4.1. COND - Conditional select one field

#### Synopsis

```
<operator> ifile1 ifile2 ofile
```

#### Description

This module conditional selects field elements from `ifile2` and writes them to `ofile`. The fields in `ifile1` are handled as a mask. A value not equal to zero is treated as "true", zero is treated as "false".

#### Operators

**ifthen**            If then

$$o(t, x) = \begin{cases} i_2(t, x) & \text{if } i_1(t, x) \neq 0 \quad \wedge \quad i_1(t, x) \neq \text{miss} \\ \text{miss} & \text{if } i_1(t, x) = 0 \quad \vee \quad i_1(t, x) = \text{miss} \end{cases}$$

**ifnotthen**        If not then

$$o(t, x) = \begin{cases} i_2(t, x) & \text{if } i_1(t, x) = 0 \quad \wedge \quad i_1(t, x) \neq \text{miss} \\ \text{miss} & \text{if } i_1(t, x) \neq 0 \quad \vee \quad i_1(t, x) = \text{miss} \end{cases}$$

#### Example

To select all field elements of `ifile2` if the corresponding field element of `ifile1` is greater than 0, use:

```
cdo ifthen ifile1 ifile2 ofile
```

### 2.4.2. COND2 - Conditional select two fields

#### Synopsis

```
ifthenelse ifile1 ifile2 ifile3 ofile
```

#### Description

This operator conditional selects field elements from `ifile2` or `ifile3` and writes them to `ofile`. The fields in `ifile1` are handled as a mask. A value not equal to zero is treated as "true", zero is treated as "false".

$$o(t, x) = \begin{cases} i_2(t, x) & \text{if } i_1(t, x) \neq 0 \quad \wedge \quad i_1(t, x) \neq \text{miss} \\ i_3(t, x) & \text{if } i_1(t, x) = 0 \quad \wedge \quad i_1(t, x) \neq \text{miss} \\ \text{miss} & \text{if } i_1(t, x) = \text{miss} \end{cases}$$

#### Example

To select all field elements of `ifile2` if the corresponding field element of `ifile1` is greater than 0 and from `ifile3` otherwise, use:

```
cdo ifthenelse ifile1 ifile2 ifile3 ofile
```

### 2.4.3. CONDC - Conditional select a constant

#### Synopsis

`<operator>,c ifile ofile`

#### Description

This module creates fields with a constant value or missing value. The fields in `ifile1` are handled as a mask. A value not equal to zero is treated as "true", zero is treated as "false".

#### Operators

<b>ifthenc</b>	If then constant
	$o(t, x) = \begin{cases} c & \text{if } i(t, x) \neq 0 \wedge i(t, x) \neq \text{miss} \\ \text{miss} & \text{if } i(t, x) = 0 \vee i(t, x) = \text{miss} \end{cases}$
<b>ifnotthenc</b>	If not then constant
	$o(t, x) = \begin{cases} c & \text{if } i(t, x) = 0 \wedge i(t, x) \neq \text{miss} \\ \text{miss} & \text{if } i(t, x) \neq 0 \vee i(t, x) = \text{miss} \end{cases}$

#### Parameter

<code>c</code>	FLOAT	Constant
----------------	-------	----------

#### Example

To create fields with the constant value 7 if the corresponding field element of `ifile` is greater than 0, use:

```
cdo ifthenc,7 ifile ofile
```

## 2.5. Comparison

This section contains modules to compare datasets. The resulting field is a mask with 1 if the comparison is true and 0 if the comparison is false.

Here is a short overview of all operators in this section:

<code>eq</code>	Equal
<code>ne</code>	Not equal
<code>le</code>	Less equal
<code>lt</code>	Less than
<code>ge</code>	Greater equal
<code>gt</code>	Greater than
<code>eqc</code>	Equal constant
<code>nec</code>	Not equal constant
<code>lec</code>	Less equal constant
<code>ltc</code>	Less then constant
<code>gec</code>	Greater equal constant
<code>gtc</code>	Greater then constant

### 2.5.1. COMP - Comparison of two fields

#### Synopsis

```
<operator> ifile1 ifile2 ofile
```

#### Description

This module compares two datasets field by field. The resulting field is a mask with 1 if the comparison is true and 0 if the comparison is false. The type of the comparison depends on the actual operator.

#### Operators

<b>eq</b>	Equal	$o(t, x) = \begin{cases} 1 & \text{if } i_1(t, x) = i_2(t, x) \wedge i_1(t, x), i_2(t, x) \neq \text{miss} \\ 0 & \text{if } i_1(t, x) \neq i_2(t, x) \wedge i_1(t, x), i_2(t, x) \neq \text{miss} \\ \text{miss} & \text{if } i_1(t, x) = \text{miss} \vee i_2(t, x) = \text{miss} \end{cases}$
<b>ne</b>	Not equal	$o(t, x) = \begin{cases} 1 & \text{if } i_1(t, x) \neq i_2(t, x) \wedge i_1(t, x), i_2(t, x) \neq \text{miss} \\ 0 & \text{if } i_1(t, x) = i_2(t, x) \wedge i_1(t, x), i_2(t, x) \neq \text{miss} \\ \text{miss} & \text{if } i_1(t, x) = \text{miss} \vee i_2(t, x) = \text{miss} \end{cases}$
<b>le</b>	Less equal	$o(t, x) = \begin{cases} 1 & \text{if } i_1(t, x) \leq i_2(t, x) \wedge i_1(t, x), i_2(t, x) \neq \text{miss} \\ 0 & \text{if } i_1(t, x) > i_2(t, x) \wedge i_1(t, x), i_2(t, x) \neq \text{miss} \\ \text{miss} & \text{if } i_1(t, x) = \text{miss} \vee i_2(t, x) = \text{miss} \end{cases}$
<b>lt</b>	Less than	$o(t, x) = \begin{cases} 1 & \text{if } i_1(t, x) < i_2(t, x) \wedge i_1(t, x), i_2(t, x) \neq \text{miss} \\ 0 & \text{if } i_1(t, x) \geq i_2(t, x) \wedge i_1(t, x), i_2(t, x) \neq \text{miss} \\ \text{miss} & \text{if } i_1(t, x) = \text{miss} \vee i_2(t, x) = \text{miss} \end{cases}$
<b>ge</b>	Greater equal	$o(t, x) = \begin{cases} 1 & \text{if } i_1(t, x) \geq i_2(t, x) \wedge i_1(t, x), i_2(t, x) \neq \text{miss} \\ 0 & \text{if } i_1(t, x) < i_2(t, x) \wedge i_1(t, x), i_2(t, x) \neq \text{miss} \\ \text{miss} & \text{if } i_1(t, x) = \text{miss} \vee i_2(t, x) = \text{miss} \end{cases}$
<b>gt</b>	Greater than	$o(t, x) = \begin{cases} 1 & \text{if } i_1(t, x) > i_2(t, x) \wedge i_1(t, x), i_2(t, x) \neq \text{miss} \\ 0 & \text{if } i_1(t, x) \leq i_2(t, x) \wedge i_1(t, x), i_2(t, x) \neq \text{miss} \\ \text{miss} & \text{if } i_1(t, x) = \text{miss} \vee i_2(t, x) = \text{miss} \end{cases}$

#### Example

To create a mask with 1 if the elements of two fields are the same and 0 if the elements are different, use:

```
cdo eq ifile1 ifile2 ofile
```



## 2.5.2. COMPC - Comparison of a field with a constant

### Synopsis

`<operator>,c ifile ofile`

### Description

This module compares all fields of dataset with a constant. The resulting field is a mask with 1 if the comparison is true and 0 if the comparison is false. The type of the comparison depends on the actual operator.

### Operators

<b>eqc</b>	Equal constant
$o(t, x) = \begin{cases} 1 & \text{if } i(t, x) = c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ 0 & \text{if } i(t, x) \neq c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ \text{miss} & \text{if } i(t, x) = \text{miss} \quad \vee \quad c = \text{miss} \end{cases}$	
<b>nec</b>	Not equal constant
$o(t, x) = \begin{cases} 1 & \text{if } i(t, x) \neq c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ 0 & \text{if } i(t, x) = c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ \text{miss} & \text{if } i(t, x) = \text{miss} \quad \vee \quad c = \text{miss} \end{cases}$	
<b>lec</b>	Less equal constant
$o(t, x) = \begin{cases} 1 & \text{if } i(t, x) \leq c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ 0 & \text{if } i(t, x) > c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ \text{miss} & \text{if } i(t, x) = \text{miss} \quad \vee \quad c = \text{miss} \end{cases}$	
<b>ltc</b>	Less then constant
$o(t, x) = \begin{cases} 1 & \text{if } i(t, x) < c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ 0 & \text{if } i(t, x) \geq c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ \text{miss} & \text{if } i(t, x) = \text{miss} \quad \vee \quad c = \text{miss} \end{cases}$	
<b>gec</b>	Greater equal constant
$o(t, x) = \begin{cases} 1 & \text{if } i(t, x) \geq c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ 0 & \text{if } i(t, x) < c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ \text{miss} & \text{if } i(t, x) = \text{miss} \quad \vee \quad c = \text{miss} \end{cases}$	
<b>gtc</b>	Greater then constant
$o(t, x) = \begin{cases} 1 & \text{if } i(t, x) > c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ 0 & \text{if } i(t, x) \leq c \quad \wedge \quad i(t, x), c \neq \text{miss} \\ \text{miss} & \text{if } i(t, x) = \text{miss} \quad \vee \quad c = \text{miss} \end{cases}$	

### Parameter

`c`    `FLOAT`    Constant

### Example

To create a mask with 1 if the field element is greater than 273.15 and 0 if not, use:

```
cdo gtc ,273.15 ifile ofile
```

## 2.6. Modification

This section contains modules to modify the metadata, fields or part of a field in a dataset.

Here is a short overview of all operators in this section:

<b>setpartab</b>	Set parameter table
<b>setcode</b>	Set code number
<b>setname</b>	Set variable name
<b>setlevel</b>	Set level
<b>setltype</b>	Set GRIB level type
<b>setdate</b>	Set date
<b>settime</b>	Set time
<b>setday</b>	Set day
<b>setmon</b>	Set month
<b>setyear</b>	Set year
<b>settimeunits</b>	Set time units
<b>settimeaxis</b>	Set time axis
<b>setreftime</b>	Set reference time
<b>setcalendar</b>	Set calendar
<b>shifttime</b>	Shift time steps
<b>chcode</b>	Change code number
<b>chname</b>	Change variable name
<b>chlevel</b>	Change level
<b>chlevelc</b>	Change level of one code
<b>chlevelv</b>	Change level of one variable
<b>setgrid</b>	Set grid
<b>setgridtype</b>	Set grid type
<b>setzaxis</b>	Set zaxis
<b>setgatt</b>	Set global attribute
<b>setgatts</b>	Set global attributes
<b>invertlat</b>	Invert latitude
<b>invertlon</b>	Invert longitude
<b>invertlatdes</b>	Invert latitude description
<b>invertlondes</b>	Invert longitude description
<b>invertlatdata</b>	Invert latitude data
<b>invertlondata</b>	Invert longitude data
<b>maskregion</b>	Mask regions
<b>masklonlatbox</b>	Mask a longitude/latitude box
<b>maskindexbox</b>	Mask an index box
<b>setclonlatbox</b>	Set a longitude/latitude box to constant
<b>setcindexbox</b>	Set an index box to constant
<b>enlarge</b>	Enlarge fields
<b>setmissval</b>	Set a new missing value
<b>setctomiss</b>	Set constant to missing value
<b>setmisstoc</b>	Set missing value to constant
<b>setrtomiss</b>	Set range to missing value

## 2.6.1. SET - Set field info

### Synopsis

```
setpartab,table ifile ofile
setcode,code ifile ofile
setname,name ifile ofile
setlevel,level ifile ofile
setltype,ltype ifile ofile
```

### Description

This module sets some field information. Depending on the actual operator the parameter table, code number, variable name or level is set.

### Operators

<b>setpartab</b>	Set parameter table Sets the parameter table for all variables.
<b>setcode</b>	Set code number Sets the code number for all variables to the same given value.
<b>setname</b>	Set variable name Sets the name of the first variable.
<b>setlevel</b>	Set level Sets the first level of all variables.
<b>setltype</b>	Set GRIB level type Sets the GRIB level type of all variables.

### Parameter

<i>table</i>	STRING	Parameter table file or name
<i>code</i>	INTEGER	Code number
<i>name</i>	STRING	Variable name
<i>level</i>	FLOAT	New level
<i>ltype</i>	INTEGER	GRIB level type

### Example

To assign the parameter table echam5 to the input dataset use:

```
cdo setpartab ,echam5 ifile ofile
```

## 2.6.2. SETTIME - Set time

### Synopsis

```

setdate,date ifile ofile
settime,time ifile ofile
setday,day ifile ofile
setmon,month ifile ofile
setyear,year ifile ofile
settunits,units ifile ofile
settaxis,date,time[,inc] ifile ofile
setreftime,date,time ifile ofile
setcalendar,calendar ifile ofile
shifttime,sval ifile ofile

```

### Description

This module sets the time axis or part of the time axis. Which part of the time axis is overwritten depends on the actual operator.

### Operators

<b>setdate</b>	Set date Sets the date in every time step to the same given value.
<b>settime</b>	Set time Sets the time in every time step to the same given value.
<b>setday</b>	Set day Sets the day in every time step to the same given value.
<b>setmon</b>	Set month Sets the month in every time step to the same given value.
<b>setyear</b>	Set year Sets the year in every time step to the same given value.
<b>settunits</b>	Set time units Sets the base units of a relative time axis.
<b>settaxis</b>	Set time axis Sets the time axis.
<b>setreftime</b>	Set reference time Sets the reference time of a relative time axis.
<b>setcalendar</b>	Set calendar Sets the calendar of a relative time axis.
<b>shifttime</b>	Shift time steps Shifts all time steps by the parameter sval.

## Parameter

<i>day</i>	INTEGER	Value of the new day
<i>month</i>	INTEGER	Value of the new month
<i>year</i>	INTEGER	Value of the new year
<i>units</i>	STRING	Base units of the time axis (minutes, hours, days, months, years)
<i>date</i>	STRING	Date (format YYYY-MM-DD)
<i>time</i>	STRING	Time (format HH:MM)
<i>inc</i>	STRING	Optional increment (minutes, hours, days, months, years) [default: 0hour]
<i>calendar</i>	STRING	Calendar (standard, 360days, 365days, 366days)
<i>sval</i>	STRING	Shift value (e.g. -3hour)

## Example

To set the time axis to 1987-01-16 12:00 with an increment of one month for each time step use:

```
cdo settaxis,1987-01-16,12:00,1mon ifile ofile
```

Result of 'cdo showdate ofile' for a dataset with 12 time steps:

```
1987-01-16 1987-02-16 1987-03-16 1987-04-16 1987-05-16 1987-06-16 \
1987-07-16 1987-08-16 1987-09-16 1987-10-16 1987-11-16 1987-12-16
```

To shift this time axis by -15 days use:

```
cdo shifttime,-15days ifile ofile
```

Result of 'cdo showdate ofile':

```
1987-01-01 1987-02-01 1987-03-01 1987-04-01 1987-05-01 1987-06-01 \
1987-07-01 1987-08-01 1987-09-01 1987-10-01 1987-11-01 1987-12-01
```

### 2.6.3. CHANGE - Change field header

#### Synopsis

```

chcode,oldcode,newcode[,...] ifile ofile
chname,ovar,nvar,... ifile ofile
chlevel,oldlev,newlev,... ifile ofile
chlevelc,code,oldlev,newlev ifile ofile
chlevelv,var,oldlev,newlev ifile ofile

```

#### Description

This module reads fields from **ifile**, changes some header values and writes the results to **ofile**. The kind of changes depends on the actual operator.

#### Operators

<b>chcode</b>	Change code number Changes some user given code numbers to new user given values.
<b>chname</b>	Change variable name Changes some user given variable names to new user given names.
<b>chlevel</b>	Change level Changes some user given levels to new user given values.
<b>chlevelc</b>	Change level of one code Changes one level of a user given code number.
<b>chlevelv</b>	Change level of one variable Changes one level of a user given variable.

#### Parameter

<i>code</i>	INTEGER	Code number
<i>oldcode,newcode</i> ,...	INTEGER	Pairs of old and new code numbers
<i>var</i>	STRING	Variable name
<i>ovar,nvar</i> ,...	STRING	Pairs of old and new variable names
<i>oldlev</i>	FLOAT	Old level
<i>newlev</i>	FLOAT	New level
<i>oldlev,newlev</i> ,...	FLOAT	Pairs of old and new levels

#### Example

To change the code number 98 to 179 and 99 to 211 use:

```
cdo chcode,98,179,99,211 ifile ofile
```

## 2.6.4. SETGRID - Set grid type

### Synopsis

```
setgrid,grid ifile ofile
setgridtype,gridtype ifile ofile
```

### Description

This module sets the grid description of all fields with the same grid size as the new grid.

### Operators

<b>setgrid</b>	Set grid Sets the grid description of all fields.
<b>setgridtype</b>	Set grid type Sets the grid type of all grids to a user given value.

### Parameter

<i>grid</i>	STRING	Target grid description file or name
<i>gridtype</i>	STRING	Target grid type (curvilinear or cell)

### Example

Assumed a dataset has fields with 2048 gridpoints without or with wrong grid description. To set the grid description of all input fields to a T21 gaussian grid (2048 gridpoints) use:

```
cdo setgrid ,t21grid ifile ofile
```

## 2.6.5. SETZAXIS - Set zaxis type

### Synopsis

```
setzaxis,zaxis ifile ofile
```

### Description

This operator sets the zaxis description of all variables with the same number of level as the new zaxis.

### Parameter

<i>zaxis</i>	STRING	Zaxis description file or name of the target zaxis
--------------	--------	--

## 2.6.6. SETGATT - Set global attribute

### Synopsis

```
setgatt,attname,attstring ifile ofile
```

```
setgatts,attfile ifile ofile
```

### Description

This module sets global text attributes of a dataset. Depending on the actual operator the attributes are read from a file or can be specified by a parameter.

### Operators

<b>setgatt</b>	Set global attribute Sets one user defined global text attribute.
<b>setgatts</b>	Set global attributes Sets user defined global text attributes. The name and text of the global attributes are read from a file.

### Parameter

<i>attname,attstring</i>	STRING	Name and text of the global attribute (without spaces!)
<i>attfile</i>	STRING	File name which contains global text attributes

### Note

From the supported data formats only netCDF can work with global attributes.

### Example

To set the global text attribute "myatt" to "myattcontents" in a netCDF file use:

```
cdo setgatt,myatt,myattcontents ifile ofile
```

Result of 'ncdump -h ofile':

```
netcdf ofile {
dimensions: ...

variables: ...

// global attributes:
           :myatt = "myattcontents" ;
}
```



## 2.6.7. INVERT - Invert fields

### Synopsis

```
<operator> ifile ofile
```

### Description

This module inverts 2D fields on a rectangular grid. Depending on the actual operator the field, only the data or only the grid description is inverted.

### Operators

<b>invertlat</b>	Invert latitude Inverts the latitude of a field.
<b>invertlon</b>	Invert longitude Inverts the longitude of a field.
<b>invertlatdes</b>	Invert latitude description Inverts only the latitude description of a field.
<b>invertlondes</b>	Invert longitude description Inverts only the longitude description of a field.
<b>invertlatdata</b>	Invert latitude data Inverts only the latitude data of a field.
<b>invertlondata</b>	Invert longitude data Inverts only the longitude data of a field.

### Example

To invert the latitudes of a 2D field from N->S to S->N, use:

```
cdo invertlat ifile ofile
```

## 2.6.8. MASKREGION - Mask regions

### Synopsis

```
maskregion,regions ifile ofile
```

### Description

Masks different regions of a field. The elements inside a region are untouched, the elements outside are set to missing value. All input fields must have the same horizontal grid. The user has to give ASCII formatted files with different regions. A region is defined by 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 &.

### Parameter

<i>regions</i>	STRING	Comma separated list of ASCII formatted files with different regions
----------------	--------	--

### Example

To mask the region with the longitudes from 120E to 90W and latitudes from 20N to 20S on all input fields, use:

```
cdo maskregion,myregion ifile ofile
```

For this example the polygon description file `myregion` must contain the following four coordinates:

```
120 20
120 -20
270 -20
270 20
```

## 2.6.9. MASKBOX - Mask a box

### Synopsis

```
masklonlatbox,lon1,lon2,lat1,lat2 ifile ofile
```

```
maskindexbox,idx1,idx2,idy1,idy2 ifile ofile
```

### Description

Masks a box of the rectangular understood field. The elements inside the box are untouched, the elements outside are set to missing value. All input fields must have the same horizontal grid.

### Operators

<b>masklonlatbox</b>	Mask a longitude/latitude box Masks a longitude/latitude box. The user has to give the longitudes and latitudes of the edges of the box.
<b>maskindexbox</b>	Mask an index box Masks an index box. The user has to give the indexes of the edges of the box. The index of the left edge may be greater then that of the right edge.

### Parameter

<i>lon1</i>	FLOAT	Western longitude
<i>lon2</i>	FLOAT	Eastern longitude
<i>lat1</i>	FLOAT	Southern or northern latitude
<i>lat2</i>	FLOAT	Northern or southern latitude
<i>idx1</i>	INTEGER	Index of first longitude
<i>idx2</i>	INTEGER	Index of last longitude
<i>idy1</i>	INTEGER	Index of first latitude
<i>idy2</i>	INTEGER	Index of last latitude

### Example

To mask the region with the longitudes from 120E to 90W and latitudes from 20N to 20S on all input fields, use:

```
cdo masklonlatbox,120,-90,20,-20 ifile ofile
```

If the input dataset has fields on a T21 gaussian grid, the same box can be masked with [maskindexbox](#) by:

```
cdo maskindexbox,23,48,13,20 ifile ofile
```

## 2.6.10. SETBOX - Set a box to constant

### Synopsis

```
setclonlatbox,c,lon1,lon2,lat1,lat2 ifile ofile
```

```
setcindexbox,c,idx1,idx2,idy1,idy2 ifile ofile
```

### Description

Sets a box of the rectangular 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 must have the same horizontal grid.

### Operators

<b>setclonlatbox</b>	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.
<b>setcindexbox</b>	Set an index box to constant Sets the values of an index box to a constant value. The user has to give the indexes of the edges of the box. The index of the left edge may be greater then that of the right edge.

### Parameter

<i>c</i>	FLOAT	Constant
<i>lon1</i>	FLOAT	Western longitude
<i>lon2</i>	FLOAT	Eastern longitude
<i>lat1</i>	FLOAT	Southern or northern latitude
<i>lat2</i>	FLOAT	Northern or southern latitude
<i>idx1</i>	INTEGER	Index of first longitude
<i>idx2</i>	INTEGER	Index of last longitude
<i>idy1</i>	INTEGER	Index of first latitude
<i>idy2</i>	INTEGER	Index of last latitude

### Example

To set all values in the region with the longitudes from 120E to 90W and latitudes from 20N to 20S to the constant value -1.23, use:

```
cdo setclonlatbox,-1.23,120,-90,20,-20 ifile ofile
```

If the input dataset has fields on a T21 gaussian grid, the same box can be set with [setcindexbox](#) by:

```
cdo setcindexbox,-1.23,23,48,13,20 ifile ofile
```

## 2.6.11. ENLARGE - Enlarge fields

### Synopsis

```
enlarge,grid ifile ofile
```

### Description

Enlarge all fields of `ifile` to a user given grid. Normally only the last field element is used for the enlargement. If however the input and output grid are rectangular, 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` must be 1 and the `xsize` of both grids must have the same size.

### Parameter

<i>grid</i>	STRING	Target grid description file or name
-------------	--------	--------------------------------------

### Example

Assumed you want to add two datasets. The first dataset is on a T21 grid (2048 field elements) and the second dataset is only a global mean (1 field element). Before you can add these two datasets the second dataset must be enlarged to the grid size of the first dataset:

```
cdo enlarge ,t21grid ifile2 tmpfile
cdo add ifile1 tmpfile ofile
```

Or shorter using operator piping:

```
cdo add ifile1 -enlarge ,t21grid ifile2 ofile
```

## 2.6.12. SETMISS - Set missing value

### Synopsis

```
setmissval,miss ifile ofile
setctomiss,c ifile ofile
setmisstoc,c ifile ofile
setrtomiss,rmin,rmax ifile ofile
```

### 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 actual operator.

### Operators

**setmissval** Set a new missing value

$$o(t, x) = \begin{cases} \text{miss} & \text{if } i(t, x) = \text{miss} \\ i(t, x) & \text{if } i(t, x) \neq \text{miss} \end{cases}$$

**setctomiss** Set constant to missing value

$$o(t, x) = \begin{cases} \text{miss} & \text{if } i(t, x) = c \\ i(t, x) & \text{if } i(t, x) \neq c \end{cases}$$

**setmisstoc** Set missing value to constant

$$o(t, x) = \begin{cases} c & \text{if } i(t, x) = \text{miss} \\ i(t, x) & \text{if } i(t, x) \neq \text{miss} \end{cases}$$

**setrtomiss** Set range to missing value

$$o(t, x) = \begin{cases} \text{miss} & \text{if } i(t, x) \geq rmin \wedge i(t, x) \leq rmax \\ i(t, x) & \text{if } i(t, x) < rmin \vee i(t, x) > rmax \end{cases}$$

### Parameter

<i>miss</i>	FLOAT	New missing value
<i>c</i>	FLOAT	Constant
<i>rmin</i>	FLOAT	Lower bound
<i>rmax</i>	FLOAT	Upper bound

### Example

Assume an input dataset has one field with the temperature in the range from 246 to 304 Kelvin. To set all values below 273.15 Kelvin to missing value use:

```
cdo setrtomiss ,0,273.15 ifile ofile
```

Result of 'cdo info ifile':

-1 :	Date	Time	Code	Level	Size	Miss :	Minimum	Mean	Maximum
1 :	1987-12-31	12:00	139	0	2048	0 :	246.27	276.75	303.71

Result of 'cdo info ofile':

-1 :	Date	Time	Code	Level	Size	Miss :	Minimum	Mean	Maximum
1 :	1987-12-31	12:00	139	0	2048	871 :	273.16	287.08	303.71

## 2.7. Arithmetic

This section contains modules to arithmetically process datasets.

Here is a short overview of all operators in this section:

<b>expr</b>	Evaluate expressions
<b>exprf</b>	Evaluate expressions from script file
<b>abs</b>	Absolute value
<b>int</b>	Integer value
<b>nint</b>	Nearest integer value
<b>sqr</b>	Square
<b>sqrt</b>	Square root
<b>exp</b>	Exponential
<b>ln</b>	Natural logarithm
<b>log10</b>	Base 10 logarithm
<b>sin</b>	Sine
<b>cos</b>	Cosine
<b>tan</b>	Tangent
<b>asin</b>	Arc sine
<b>acos</b>	Arc cosine
<b>atan</b>	Arc tangent
<b>addc</b>	Add a constant
<b>subc</b>	Subtract a constant
<b>mulc</b>	Multiply with a constant
<b>divc</b>	Divide by a constant
<b>add</b>	Add two fields
<b>sub</b>	Subtract two fields
<b>mul</b>	Multiply two fields
<b>div</b>	Divide two fields
<b>min</b>	Minimum of two fields
<b>max</b>	Maximum of two fields
<b>atan2</b>	Arc tangent of two fields
<b>monadd</b>	Add monthly time series
<b>monsub</b>	Subtract monthly time series
<b>monmul</b>	Multiply monthly time series
<b>monddiv</b>	Divide monthly time series
<b>ymonadd</b>	Add multi-year monthly time series
<b>ymonsub</b>	Subtract multi-year monthly time series
<b>ymonmul</b>	Multiply multi-year monthly time series
<b>ymonddiv</b>	Divide multi-year monthly time series
<b>muldpm</b>	Multiply with days per month
<b>divdpm</b>	Divide by days per month
<b>muldpy</b>	Multiply with days per year
<b>divdpy</b>	Divide by days per year

## 2.7.1. EXPR - Evaluate expressions

### Synopsis

```
expr,instr ifile ofile
exprf,filename ifile ofile
```

### Description

This module arithmetically processes every time step of the input dataset. Each individual assignment statement must end with a semi-colon. The basic arithmetic operations addition +, subtraction −, multiplication \*, division / and exponentiation ^ can be used. The following intrinsic functions are available:

<code>sqrt(x)</code>	Square Root of x
<code>exp(x)</code>	Exponential of x
<code>log(x)</code>	Natural logarithm of x
<code>log10(x)</code>	Base 10 logarithm of x
<code>sin(x)</code>	Sine of x, where x is specified in radians
<code>cos(x)</code>	Cosine of x, where x is specified in radians
<code>tan(x)</code>	Tangent of x, where x is specified in radians
<code>asin(x)</code>	Arc-sine of x, where x is specified in radians
<code>acos(x)</code>	Arc-cosine of x, where x is specified in radians
<code>atan(x)</code>	Arc-tangent of x, where x is specified in radians

### Operators

<b>expr</b>	Evaluate expressions The processing instructions are read from the parameter.
<b>exprf</b>	Evaluate expressions from script file Contrary to <a href="#">expr</a> the processing instructions are read from a file.

### Parameter

<i>instr</i>	STRING	Processing instructions (without spaces!)
<i>filename</i>	STRING	File with processing instructions

### Example

Assume an input dataset contains at least the variables 'aprl', 'aprc' and 'ts'. To create a new variable 'var1' with the sum of 'aprl' and 'aprc' and a variable 'var2' which convert the temperature 'ts' from Kelvin to Celsius use:

```
cdo expr, 'var1=aprl+aprc; var2=ts - 273.15;' ifile ofile
```

The same example, but the instructions are read from a file:

```
cdo exprf, myexpr ifile ofile
```

The file `myexpr` contains:

```
var1 = aprl + aprc;
var2 = ts - 273.15;
```



## 2.7.2. MATH - Mathematical functions

### Synopsis

`<operator> ifile ofile`

### Description

This module contains some standard mathematical functions. All trigonometric functions calculate with radians.

### Operators

<b>abs</b>	Absolute value $o(t, x) = \text{abs}(i(t, x))$
<b>int</b>	Integer value $o(t, x) = \text{int}(i(t, x))$
<b>nint</b>	Nearest integer value $o(t, x) = \text{nint}(i(t, x))$
<b>sqr</b>	Square $o(t, x) = i(t, x)^2$
<b>sqrt</b>	Square root $o(t, x) = \sqrt{i(t, x)}$
<b>exp</b>	Exponential $o(t, x) = e^{i(t, x)}$
<b>ln</b>	Natural logarithm $o(t, x) = \ln(i(t, x))$
<b>log10</b>	Base 10 logarithm $o(t, x) = \log_{10}(i(t, x))$
<b>sin</b>	Sine $o(t, x) = \sin(i(t, x))$
<b>cos</b>	Cosine $o(t, x) = \cos(i(t, x))$
<b>tan</b>	Tangent $o(t, x) = \tan(i(t, x))$
<b>asin</b>	Arc sine $o(t, x) = \arcsin(i(t, x))$
<b>acos</b>	Arc cosine $o(t, x) = \arccos(i(t, x))$
<b>atan</b>	Arc tangent $o(t, x) = \arctan(i(t, x))$

### Example

To calculate the square root for all field elements use:

```
cdo sqrt ifile ofile
```

### 2.7.3. ARITHC - Arithmetic with a constant

#### Synopsis

`<operator>,c ifile ofile`

#### Description

This module performs simple arithmetic with all field elements of a dataset and a constant. The header and date information in `ofile` is the same as in `ifile`.

#### Operators

<b>addc</b>	Add a constant $o(t, x) = i(t, x) + c$
<b>subc</b>	Subtract a constant $o(t, x) = i(t, x) - c$
<b>mulc</b>	Multiply with a constant $o(t, x) = i(t, x) * c$
<b>divc</b>	Divide by a constant $o(t, x) = i(t, x) / c$

#### Parameter

<code>c</code>	FLOAT	Constant
----------------	-------	----------

#### Example

To sum all input fields with the constant -273.15 use:

```
cdo addc,-273.15 ifile ofile
```

## 2.7.4. ARITH - Arithmetic on two datasets

### Synopsis

```
<operator> ifile1 ifile2 ofile
```

### Description

This module performs simple arithmetic of two datasets. The header and date information in **ofile** is the same as in **ifile1**.

### Operators

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

### Example

To sum all fields of the first input file with the corresponding fields of the second input file use:

```
cdo add ifile1 ifile2 ofile
```

## 2.7.5. MONARITH - Monthly arithmetic

### Synopsis

```
<operator> ifile1 ifile2 ofile
```

### Description

This module performs simple arithmetic of a time series and a time step with the same month and year. For each field in `ifile1` the corresponding field of the time step in `ifile2` with the same month and year is used. The header information in `ifile1` must be the same as in `ifile2`. Usually `ifile2` is generated by a call of the module [MONSTAT](#).

### Operators

<b>monadd</b>	Add monthly time series Adds a time series and a monthly time series.
<b>monsub</b>	Subtract monthly time series Subtracts a time series and a monthly time series.
<b>monmul</b>	Multiply monthly time series Multiplies a time series and a monthly time series.
<b>monddiv</b>	Divide monthly time series Divides a time series and a monthly time series.

### Example

To subtract a monthly time average from a time series, use:

```
cdo monsub ifile -monavg ifile ofile
```

## 2.7.6. YMONARITH - Multi-year monthly arithmetic

### Synopsis

```
<operator> ifile1 ifile2 ofile
```

### Description

This module performs simple arithmetic of a time series and a time step with the same month of year. For each field in `ifile1` the corresponding field of the time step in `ifile2` with the same month of year is used. The header information in `ifile1` must be the same as in `ifile2`. Usually `ifile2` is generated by a call of the module [YMONSTAT](#).

### Operators

<b>ymonadd</b>	Add multi-year monthly time series Adds a time series and a multi-year monthly time series.
<b>ymonsub</b>	Subtract multi-year monthly time series Subtracts a time series and a multi-year monthly time series.
<b>ymonmul</b>	Multiply multi-year monthly time series Multiplies a time series and a multi-year monthly time series.
<b>ymonddiv</b>	Divide multi-year monthly time series Divides a time series and a multi-year monthly time series.

### Example

To subtract a multi-year monthly time average from a time series, use:

```
cdo ymonsub ifile -ymonavg ifile ofile
```

## 2.7.7. ARITHDAYS - Arithmetic with days

### Synopsis

`<operator> ifile ofile`

### Description

This module multiplies or divides each time step of a dataset with the corresponding days per month or days per year.

### Operators

<b>muldpm</b>	Multiply with days per month $o(t, x) = i(t, x) * days\_per\_month$
<b>divdpm</b>	Divide by days per month $o(t, x) = i(t, x) / days\_per\_month$
<b>muldpy</b>	Multiply with days per year $o(t, x) = i(t, x) * days\_per\_year$
<b>divdpy</b>	Divide by days per year $o(t, x) = i(t, x) / days\_per\_year$

### Example

Assume an input dataset is a monthly mean time series. To compute the yearly mean from the correct weighted monthly mean use:

```
cdo muldpm ifile tmpfile1
cdo yearavg tmpfile1 tmpfile2
cdo mulc,12 -divdpy tmpfile2 ofile
```

Or all in one command line:

```
cdo mulc,12 -divdpy -yearavg -muldpm ifile ofile
```

## 2.8. Statistical values

This section contains modules to compute statistical values of datasets. In this program there is the different notion of "mean" and "average" to distinguish two different kinds of treatment of missing values. While computing the mean, only the not missing values are considered to belong to the sample with the side effect of a probably reduced sample size. Computing the average is just adding the sample members and divide the result by the sample size. For example, the mean of 1, 2, miss and 3 is  $(1+2+3)/3 = 2$ , whereas the average is  $(1+2+miss+3)/4 = miss/4 = miss$ . If there are no missing values in the sample, the average and the mean are identical.

In this section the abbreviations as in the following table are used:

<b>sum</b>	$\sum_{i=1}^n x_i$
<b>mean</b> resp. <b>avg</b>	$n^{-1} \sum_{i=1}^n x_i$
<b>mean</b> resp. <b>avg</b> weighted by $\{w_i, i = 1, \dots, n\}$	$\left( \sum_{j=1}^n w_j \right)^{-1} \sum_{i=1}^n w_i x_i$
Variance <b>var</b>	$n^{-1} \sum_{i=1}^n (x_i - \bar{x})^2$
<b>var</b> weighted by $\{w_i, i = 1, \dots, n\}$	$\left( \sum_{j=1}^n w_j \right)^{-1} \sum_{i=1}^n w_i \left( x_i - \left( \sum_{j=1}^n w_j \right)^{-1} \sum_{j=1}^n w_j x_j \right)^2$
Standard deviation <b>std</b>	$\sqrt{n^{-1} \sum_{i=1}^n (x_i - \bar{x})^2}$
<b>std</b> weighted by $\{w_i, i = 1, \dots, n\}$	$\sqrt{\left( \sum_{j=1}^n w_j \right)^{-1} \sum_{i=1}^n w_i \left( x_i - \left( \sum_{j=1}^n w_j \right)^{-1} \sum_{j=1}^n w_j x_j \right)^2}$

Here is a short overview of all operators in this section:

<b>ensmin</b>	Ensemble minimum
<b>ensmax</b>	Ensemble maximum
<b>enssum</b>	Ensemble sum
<b>ensmean</b>	Ensemble mean
<b>ensavg</b>	Ensemble average
<b>ensvar</b>	Ensemble variance
<b>ensstd</b>	Ensemble standard deviation
<b>enspctl</b>	Ensemble percentiles

<b>fldmin</b>	Field minimum
<b>fldmax</b>	Field maximum
<b>fldsum</b>	Field sum
<b>fldmean</b>	Field mean
<b>fldavg</b>	Field average
<b>fldvar</b>	Field variance
<b>fldstd</b>	Field standard deviation
<b>fldpctl</b>	Field percentiles
<b>zonmin</b>	Zonal minimum
<b>zonmax</b>	Zonal maximum
<b>zonsum</b>	Zonal sum
<b>zonmean</b>	Zonal mean
<b>zonavg</b>	Zonal average
<b>zonvar</b>	Zonal variance
<b>zonstd</b>	Zonal standard deviation
<b>zonpctl</b>	Zonal percentiles
<b>mermin</b>	Meridional minimum
<b>mermax</b>	Meridional maximum
<b>mersum</b>	Meridional sum
<b>mermean</b>	Meridional mean
<b>meravg</b>	Meridional average
<b>mervar</b>	Meridional variance
<b>merstd</b>	Meridional standard deviation
<b>merpctl</b>	Meridional percentiles
<b>vertmin</b>	Vertical minimum
<b>vertmax</b>	Vertical maximum
<b>vertsum</b>	Vertical sum
<b>vertmean</b>	Vertical mean
<b>vertavg</b>	Vertical average
<b>vertvar</b>	Vertical variance
<b>vertstd</b>	Vertical standard deviation
<b>timselmin</b>	Time range minimum
<b>timselmax</b>	Time range maximum
<b>timselsum</b>	Time range sum
<b>timselmean</b>	Time range mean
<b>timselavg</b>	Time range average
<b>timselvar</b>	Time range variance
<b>timselstd</b>	Time range standard deviation
<b>timselpctl</b>	Time range percentiles
<b>runmin</b>	Running minimum
<b>runmax</b>	Running maximum
<b>runsum</b>	Running sum
<b>runmean</b>	Running mean
<b>runavg</b>	Running average
<b>runvar</b>	Running variance
<b>runstd</b>	Running standard deviation
<b>runpctl</b>	Running percentiles



<b>timmin</b>	Time minimum
<b>timmax</b>	Time maximum
<b>timsum</b>	Time sum
<b>timmean</b>	Time mean
<b>timavg</b>	Time average
<b>timvar</b>	Time variance
<b>timstd</b>	Time standard deviation
<b>timctl</b>	Time percentiles
<b>hourmin</b>	Hourly minimum
<b>hourmax</b>	Hourly maximum
<b>hoursum</b>	Hourly sum
<b>hourmean</b>	Hourly mean
<b>houravg</b>	Hourly average
<b>hourvar</b>	Hourly variance
<b>hourstd</b>	Hourly standard deviation
<b>hourctl</b>	Hourly percentiles
<b>daymin</b>	Daily minimum
<b>daymax</b>	Daily maximum
<b>daysum</b>	Daily sum
<b>daymean</b>	Daily mean
<b>dayavg</b>	Daily average
<b>dayvar</b>	Daily variance
<b>daystd</b>	Daily standard deviation
<b>dayctl</b>	Daily percentiles
<b>monmin</b>	Monthly minimum
<b>monmax</b>	Monthly maximum
<b>monsum</b>	Monthly sum
<b>monmean</b>	Monthly mean
<b>monavg</b>	Monthly average
<b>monvar</b>	Monthly variance
<b>monstd</b>	Monthly standard deviation
<b>monctl</b>	Monthly percentiles
<b>yearmin</b>	Yearly minimum
<b>yearmax</b>	Yearly maximum
<b>yearsum</b>	Yearly sum
<b>yearmean</b>	Yearly mean
<b>yearavg</b>	Yearly average
<b>yearvar</b>	Yearly variance
<b>yearstd</b>	Yearly standard deviation
<b>yearctl</b>	Yearly percentiles
<b>seasmin</b>	Seasonal minimum
<b>seasmax</b>	Seasonal maximum
<b>seassum</b>	Seasonal sum
<b>seasmean</b>	Seasonal mean
<b>seasavg</b>	Seasonal average
<b>seasvar</b>	Seasonal variance
<b>seasstd</b>	Seasonal standard deviation

<b>seaspctl</b>	Seasonal percentiles
<b>yhourmin</b>	Multi-year hourly minimum
<b>yhourmax</b>	Multi-year hourly maximum
<b>yhoursum</b>	Multi-year hourly sum
<b>yhourmean</b>	Multi-year hourly mean
<b>yhouravg</b>	Multi-year hourly average
<b>yhourvar</b>	Multi-year hourly variance
<b>yhourstd</b>	Multi-year hourly standard deviation
<b>ydaymin</b>	Multi-year daily minimum
<b>ydaymax</b>	Multi-year daily maximum
<b>ydaysum</b>	Multi-year daily sum
<b>ydaymean</b>	Multi-year daily mean
<b>ydayavg</b>	Multi-year daily average
<b>ydayvar</b>	Multi-year daily variance
<b>ydaystd</b>	Multi-year daily standard deviation
<b>ydaypctl</b>	Multi-year daily percentiles
<b>ymonmin</b>	Multi-year monthly minimum
<b>ymonmax</b>	Multi-year monthly maximum
<b>ymonsum</b>	Multi-year monthly sum
<b>ymonmean</b>	Multi-year monthly mean
<b>ymonavg</b>	Multi-year monthly average
<b>ymonvar</b>	Multi-year monthly variance
<b>ymonstd</b>	Multi-year monthly standard deviation
<b>ymonpctl</b>	Multi-year monthly percentiles
<b>yseasmin</b>	Multi-year seasonal minimum
<b>yseasmax</b>	Multi-year seasonal maximum
<b>yseasum</b>	Multi-year seasonal sum
<b>yseasmean</b>	Multi-year seasonal mean
<b>yseasavg</b>	Multi-year seasonal average
<b>yseasvar</b>	Multi-year seasonal variance
<b>yseasstd</b>	Multi-year seasonal standard deviation
<b>yseaspctl</b>	Multi-year seasonal percentiles
<b>ydrunmin</b>	Multi-year daily running minimum
<b>ydrunmax</b>	Multi-year daily running maximum
<b>ydrunsum</b>	Multi-year daily running sum
<b>ydrunmean</b>	Multi-year daily running mean
<b>ydrunavg</b>	Multi-year daily running average
<b>ydrunvar</b>	Multi-year daily running variance
<b>ydrunstd</b>	Multi-year daily running standard deviation
<b>ydrunpctl</b>	Multi-year daily running percentiles

## 2.8.1. ENSSTAT - Statistical values over an ensemble

### Synopsis

```

ensmin ifiles ofile
ensmax ifiles ofile
enssum ifiles ofile
ensmean ifiles ofile
ensavg ifiles ofile
ensvar ifiles ofile
ensstd ifiles ofile
enspctl,p ifiles ofile

```

### Description

This module computes statistical values over an ensemble of input files. Depending on the actual operator the minimum, maximum, sum, average, variance, standard deviation, or a certain percentile over all input files is written to `ofile`. The date information for a time step in `ofile` is the date of the first input file.

### Operators

<b>ensmin</b>	Ensemble minimum $o(t, x) = \mathbf{min}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$
<b>ensmax</b>	Ensemble maximum $o(t, x) = \mathbf{max}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$
<b>enssum</b>	Ensemble sum $o(t, x) = \mathbf{sum}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$
<b>ensmean</b>	Ensemble mean $o(t, x) = \mathbf{mean}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$
<b>ensavg</b>	Ensemble average $o(t, x) = \mathbf{avg}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$
<b>ensvar</b>	Ensemble variance $o(t, x) = \mathbf{var}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$
<b>ensstd</b>	Ensemble standard deviation $o(t, x) = \mathbf{std}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$
<b>enspctl</b>	Ensemble percentiles $o(t, x) = \mathbf{pth\ percentile}\{i_1(t, x), i_2(t, x), \dots, i_n(t, x)\}$

### Parameter

*p*     INTEGER     Percentile number in 1, ..., 99

## Example

To compute the ensemble mean over 6 input files, use:

```
cdo ensmean ifile1 ifile2 ifile3 ifile4 ifile5 ifile6 ofile
```

Or shorter with filename substitution:

```
cdo ensmean ifile[1-6] ofile
```

To compute the 50th percentile (median) over 6 input files, use:

```
cdo enspctl,50 ifile1 ifile2 ifile3 ifile4 ifile5 ifile6 ofile
```

Or shorter with filename substitution:

```
cdo enspctl,50 ifile[1-6] ofile
```

## 2.8.2. FLDSTAT - Statistical values over a field

### Synopsis

```
fldmin ifile ofile
fldmax ifile ofile
fldsum ifile ofile
fldmean ifile ofile
fldavg ifile ofile
fldvar ifile ofile
fldstd ifile ofile
fldpctl,p ifile ofile
```

### Description

This module computes statistical values of the input fields. According to the actual operator the field minimum, maximum, sum, average, variance, standard deviation or a certain percentile is written to ofile.

### Operators

<b>fldmin</b>	Field minimum For every gridpoint $x_1, \dots, x_n$ of the same field, it is: $o(t, 1) = \mathbf{min}\{i(t, x'), x_1 < x' \leq x_n\}$
<b>fldmax</b>	Field maximum For every gridpoint $x_1, \dots, x_n$ of the same field, it is: $o(t, 1) = \mathbf{max}\{i(t, x'), x_1 < x' \leq x_n\}$
<b>fldsum</b>	Field sum For every gridpoint $x_1, \dots, x_n$ of the same field, it is: $o(t, 1) = \mathbf{sum}\{i(t, x'), x_1 < x' \leq x_n\}$
<b>fldmean</b>	Field mean For every gridpoint $x_1, \dots, x_n$ of the same field, it is: $o(t, 1) = \mathbf{mean}\{i(t, x'), x_1 < x' \leq x_n\}$ weighted by area weights obtained by the input field.
<b>fldavg</b>	Field average For every gridpoint $x_1, \dots, x_n$ of the same field, it is: $o(t, 1) = \mathbf{avg}\{i(t, x'), x_1 < x' \leq x_n\}$ weighted by area weights obtained by the input field.
<b>fldvar</b>	Field variance For every gridpoint $x_1, \dots, x_n$ of the same field, it is: $o(t, 1) = \mathbf{var}\{i(t, x'), x_1 < x' \leq x_n\}$ weighted by area weights obtained by the input field.
<b>fldstd</b>	Field standard deviation For every gridpoint $x_1, \dots, x_n$ of the same field, it is: $o(t, 1) = \mathbf{std}\{i(t, x'), x_1 < x' \leq x_n\}$ weighted by area weights obtained by the input field.
<b>fldpctl</b>	Field percentiles For every gridpoint $x_1, \dots, x_n$ of the same field, it is: $o(t, 1) = \mathbf{pth\ percentile}\{i(t, x'), x_1 < x' \leq x_n\}$

**Parameter**

$p$     INTEGER    Percentile number in 1, ..., 99

**Example**

To compute the field mean of all input fields, use:

```
cdo fldmean ifile ofile
```

To compute the 90th percentile of all input fields, use:

```
cdo fldpctl,90 ifile ofile
```

### 2.8.3. ZONSTAT - Zonal statistical values

#### Synopsis

```
zonmin ifile ofile
zonmax ifile ofile
zonsum ifile ofile
zonmean ifile ofile
zonavg ifile ofile
zonvar ifile ofile
zonstd ifile ofile
zonpctl,p ifile ofile
```

#### Description

This module computes zonal statistical values of the input fields. According to the actual operator the zonal minimum, maximum, sum, average, variance, standard deviation or a certain percentile is written to ofile. All input fields must have the same rectangular grid.

#### Operators

<b>zonmin</b>	Zonal minimum For every latitude the minimum over all longitudes is computed.
<b>zonmax</b>	Zonal maximum For every latitude the maximum over all longitudes is computed.
<b>zonsum</b>	Zonal sum For every latitude the sum over all longitudes is computed.
<b>zonmean</b>	Zonal mean For every latitude the mean over all longitudes is computed.
<b>zonavg</b>	Zonal average For every latitude the average over all longitudes is computed.
<b>zonvar</b>	Zonal variance For every latitude the variance over all longitudes is computed.
<b>zonstd</b>	Zonal standard deviation For every latitude the standard deviation over all longitudes is computed.
<b>zonpctl</b>	Zonal percentiles For every latitude the pth percentile over all longitudes is computed.

#### Parameter

<i>p</i>	INTEGER	Percentile number in 1, ..., 99
----------	---------	---------------------------------

**Example**

To compute the zonal mean of all input fields, use:

```
cdo zonmean ifile ofile
```

To compute the 50th meridional percentile (median) of all input fields, use:

```
cdo zonpctl,50 ifile ofile
```



## 2.8.4. MERSTAT - Meridional statistical values

### Synopsis

```

mermin ifile ofile
mermax ifile ofile
mersum ifile ofile
mermean ifile ofile
meravg ifile ofile
mervar ifile ofile
merstd ifile ofile
merpctl,p ifile ofile

```

### Description

This module computes meridional statistical values of the input fields. According to the actual operator the meridional minimum, maximum, sum, average, variance, standard deviation or a certain percentile is written to `ofile`. All input fields must have the same rectangular grid.

### Operators

<b>mermin</b>	Meridional minimum For every longitude the minimum over all latitudes is computed.
<b>mermax</b>	Meridional maximum For every longitude the maximum over all latitudes is computed.
<b>mersum</b>	Meridional sum For every longitude the sum over all latitudes is computed.
<b>mermean</b>	Meridional mean For every longitude the area weighted mean over all latitudes is computed.
<b>meravg</b>	Meridional average For every longitude the area weighted average over all latitudes is computed.
<b>mervar</b>	Meridional variance For every longitude the variance over all latitudes is computed.
<b>merstd</b>	Meridional standard deviation For every longitude the standard deviation over all latitudes is computed.
<b>merpctl</b>	Meridional percentiles For every longitude the <i>p</i> th percentile over all latitudes is computed.

### Parameter

*p*    INTEGER    Percentile number in 1, ..., 99

**Example**

To compute the meridional mean of all input fields, use:

```
cdo mermean ifile ofile
```

To compute the 50th meridional percentile (median) of all input fields, use:

```
cdo merpctl,50 ifile ofile
```

## 2.8.5. VERTSTAT - Vertical statistical values

### Synopsis

```
<operator> ifile ofile
```

### Description

This module computes statistical values over all levels of the input variables. According to actual operator the vertical minimum, maximum, sum, average, variance or standard deviation is written to ofile.

### Operators

<b>vertmin</b>	Vertical minimum For every gridpoint the minimum over all levels is computed.
<b>vertmax</b>	Vertical maximum For every gridpoint the maximum over all levels is computed.
<b>vertsum</b>	Vertical sum For every gridpoint the sum over all levels is computed.
<b>vertmean</b>	Vertical mean For every gridpoint the mean over all levels is computed.
<b>vertavg</b>	Vertical average For every gridpoint the average over all levels is computed.
<b>vertvar</b>	Vertical variance For every gridpoint the variance over all levels is computed.
<b>vertstd</b>	Vertical standard deviation For every gridpoint the standard deviation over all levels is computed.

### Example

To compute the vertical sum of all input variables, use:

```
cdo vertsum ifile ofile
```

## 2.8.6. TIMSELSTAT - Time range statistical values

### Synopsis

```
<operator>,nsets[,noffset[,nskip]] ifile ofile
```

### Description

This module computes statistical values for a selected number of time steps. According to the actual operator the average, minimum, maximum, sum, average, variance or standard deviation of the selected time steps is written to **ofile**. The date information for a time step in **ofile** is the date of the last contributing time step in **ifile**.

### Operators

<b>timselmin</b>	Time range minimum For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same selected time range, it is: $o(t, x) = \min\{i(t', x), t_1 < t' \leq t_n\}$
<b>timselmax</b>	Time range maximum For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same selected time range, it is: $o(t, x) = \max\{i(t', x), t_1 < t' \leq t_n\}$
<b>timselsum</b>	Time range sum For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same selected time range, it is: $o(t, x) = \sum\{i(t', x), t_1 < t' \leq t_n\}$
<b>timselmean</b>	Time range mean For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same selected time range, it is: $o(t, x) = \text{mean}\{i(t', x), t_1 < t' \leq t_n\}$
<b>timselavg</b>	Time range average For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same selected time range, it is: $o(t, x) = \text{avg}\{i(t', x), t_1 < t' \leq t_n\}$
<b>timselvar</b>	Time range variance For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same selected time range, it is: $o(t, x) = \text{var}\{i(t', x), t_1 < t' \leq t_n\}$
<b>timselstd</b>	Time range standard deviation For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same selected time range, it is: $o(t, x) = \text{std}\{i(t', x), t_1 < t' \leq t_n\}$

### Parameter

<i>nsets</i>	INTEGER	Number of input time steps for each output time step
<i>noffset</i>	INTEGER	Number of input time steps skipped before the first time step range (optional)
<i>nskip</i>	INTEGER	Number of input time steps skipped between time step ranges (optional)

## Example

Assume an input dataset has monthly means over several years. To compute seasonal means from monthly means the first two month must be skipped:

```
cdo timselmean,3,2 ifile ofile
```

## 2.8.7. TIMSELPCTL - Time range percentile values

### Synopsis

```
timselpctl,p,nsets[,noffset[,nskip]] ifile1 ifile2 ifile3 ofile
```

### Description

This module computes percentile values over a selected number of time steps in `ifile1`. The algorithm uses histograms with minimum and maximum bounds given in `ifile2` and `ifile3`, respectively. The default number of histogram bins is 100. The default can be overridden by setting the environment variable `CDO_PCTL_NBINS` to a different value. The files `ifile2` and `ifile3` must be the result of corresponding `timselmin` and `timselmax` operations, respectively. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

For every adjacent sequence  $t_1, \dots, t_n$  of time steps of the same selected time range, it is:

$o(t, x) = \text{pth percentile}\{i(t', x), t_1 < t' \leq t_n\}$

### Parameter

<i>p</i>	INTEGER	Percentile number in 1, ..., 99
<i>nsets</i>	INTEGER	Number of input time steps for each output time step
<i>noffset</i>	INTEGER	Number of input time steps skipped before the first time step range (optional)
<i>nskip</i>	INTEGER	Number of input time steps skipped between time step ranges (optional)

### Environment

<code>CDO_PCTL_NBINS</code>	Sets the number of histogram bins. The default number is 100.
-----------------------------	---

## 2.8.8. RUNSTAT - Running statistical values

### Synopsis

```
<operator>,nts ifile ofile
```

### Description

This module computes running statistical values over a selected number of time steps. Depending on the actual operator the minimum, maximum, sum, average, variance or standard deviation of a selected number of consecutive time steps read from **ifile** is written to **ofile**. The date information in **ofile** is the date of the middle contributing time step in **ifile**.

### Operators

<b>runmin</b>	Running minimum $o(t + (nts - 1)/2, x) = \min\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$
<b>runmax</b>	Running maximum $o(t + (nts - 1)/2, x) = \max\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$
<b>runsum</b>	Running sum $o(t + (nts - 1)/2, x) = \text{sum}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$
<b>runmean</b>	Running mean $o(t + (nts - 1)/2, x) = \text{mean}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$
<b>runavg</b>	Running average $o(t + (nts - 1)/2, x) = \text{avg}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$
<b>runvar</b>	Running variance $o(t + (nts - 1)/2, x) = \text{std}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$
<b>runstd</b>	Running standard deviation $o(t + (nts - 1)/2, x) = \text{std}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$

### Parameter

**nts**     **INTEGER**     Number of time steps

### Environment

**RUNSTAT\_DATE**     Sets the date information in ofile to the "first", "last" or "middle" contributing time step in ifile.

### Example

To compute the running mean over 9 time steps, use:

```
cdo runmean,9 ifile ofile
```

## 2.8.9. RUNPCTL - Running percentile values

### Synopsis

```
runpctl,p,nts ifile1 ofile
```

### Description

This module computes running percentiles over a selected number of time steps in **ifile1**. The date information in **ofile** is the date of the medium contributing time step in **ifile1**.  
 $o(t + (nts - 1)/2, x) = \mathbf{pth\ percentile}\{i(t, x), i(t + 1, x), \dots, i(t + nts - 1, x)\}$

### Parameter

<i>p</i>	INTEGER	Percentile number in 1, ..., 99
<i>nts</i>	INTEGER	Number of time steps

### Example

To compute the running 50th percentile (median) over 9 time steps, use:

```
cdo runpctl,50,9 ifile -runmin,9 ifile -runmax,9 ifile ofile
```

## 2.8.10. TIMSTAT - Statistical values over all time steps

### Synopsis

```
<operator> ifile ofile
```

### Description

This module computes statistical values over all time steps in `ifile`. Depending on the actual operator the minimum, maximum, sum, average, variance or standard deviation of all time steps read from `ifile` is written to `ofile`. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

### Operators

<b>timmin</b>	Time minimum $o(1, x) = \mathbf{min}\{i(t', x), t_1 < t' \leq t_n\}$
<b>timmax</b>	Time maximum $o(1, x) = \mathbf{max}\{i(t', x), t_1 < t' \leq t_n\}$
<b>timsum</b>	Time sum $o(1, x) = \mathbf{sum}\{i(t', x), t_1 < t' \leq t_n\}$
<b>timmean</b>	Time mean $o(1, x) = \mathbf{mean}\{i(t', x), t_1 < t' \leq t_n\}$
<b>timavg</b>	Time average $o(1, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$
<b>timvar</b>	Time variance $o(1, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$
<b>timstd</b>	Time standard deviation $o(1, x) = \mathbf{std}\{i(t', x), t_1 < t' \leq t_n\}$

### Example

To compute the mean over all input time steps, use:

```
cdo timmean ifile ofile
```



## 2.8.11. TIMPCTL - Percentile values over all time steps

### Synopsis

```
timpctl,p ifile1 ifile2 ifile3 ofile
```

### Description

This module computes percentiles over all time steps in **ifile1**. The algorithm uses histograms with minimum and maximum bounds given in **ifile2** and **ifile3**, respectively. The default number of histogram bins is 100. The default can be overridden by setting the environment variable **CDO\_PCTL\_NBINS** to a different value. The files **ifile2** and **ifile3** must be the result of corresponding **timmin** and **timmax** operations, respectively. The date information for a time step in **ofile** is the date of the last contributing time step in **ifile1**.

$o(1, x) = \text{pth percentile}\{i(t', x), t_1 < t' \leq t_n\}$

### Parameter

*p*     INTEGER     Percentile number in 1, ..., 99

### Environment

**CDO\_PCTL\_NBINS**     Sets the number of histogram bins. The default number is 100.

### Example

To compute the 90th percentile over all input time steps, use:

```
cdo timmin ifile minfile
cdo timmax ifile maxfile
cdo timpctl,90 ifile minfile maxfile ofile
```

Or shorter using operator piping:

```
cdo timpctl,90 ifile -timmin ifile -timmax ifile ofile
```

## 2.8.12. HOURSTAT - Hourly statistical values

### Synopsis

```
<operator> ifile ofile
```

### Description

This module computes statistical values over time steps of the same hour. Depending on the actual operator the minimum, maximum, sum, average, variance or standard deviation of time steps of the same hour is written to `ofile`. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

### Operators

<b>hourmin</b>	Hourly minimum For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same hour, it is: $o(t, x) = \mathbf{min}\{i(t', x), t_1 < t' \leq t_n\}$
<b>hourmax</b>	Hourly maximum For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same hour, it is: $o(t, x) = \mathbf{max}\{i(t', x), t_1 < t' \leq t_n\}$
<b>hoursum</b>	Hourly sum For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same hour, it is: $o(t, x) = \mathbf{sum}\{i(t', x), t_1 < t' \leq t_n\}$
<b>hourmean</b>	Hourly mean For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same hour, it is: $o(t, x) = \mathbf{mean}\{i(t', x), t_1 < t' \leq t_n\}$
<b>houravg</b>	Hourly average For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same hour, it is: $o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$
<b>hourvar</b>	Hourly variance For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same hour, it is: $o(t, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$
<b>hourstd</b>	Hourly standard deviation For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same hour, it is: $o(t, x) = \mathbf{std}\{i(t', x), t_1 < t' \leq t_n\}$

### Example

To compute the hourly mean of a time series, use:

```
cdo hourmean ifile ofile
```

## 2.8.13. HOURPCTL - Hourly percentile values

### Synopsis

```
hourpctl,p ifile1 ifile2 ifile3 ofile
```

### Description

This module computes percentiles over all time steps in `ifile1` of the same hour. The algorithm uses histograms with minimum and maximum bounds given in `ifile2` and `ifile3`, respectively. The default number of histogram bins is 100. The default can be overridden by setting the environment variable `CDO_PCTL_NBINS` to a different value. The files `ifile2` and `ifile3` must be the result of corresponding `hourmin` and `hourmax` operations, respectively. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

For every adjacent sequence  $t_1, \dots, t_n$  of time steps of the same hour, it is:

$$o(t, x) = \text{pth percentile}\{i(t', x), t_1 < t' \leq t_n\}$$

### Parameter

*p*     INTEGER     Percentile number in 1, ..., 99

### Environment

`CDO_PCTL_NBINS`     Sets the number of histogram bins. The default number is 100.

### Example

To compute the hourly 90th percentile of a time series, use:

```
cdo hourmin ifile minfile
cdo hourmax ifile maxfile
cdo hourpctl,90 ifile minfile maxfile ofile
```

Or shorter using operator piping:

```
cdo hourpctl,90 ifile -hourmin ifile -hourmax ifile ofile
```

## 2.8.14. DAYSTAT - Daily statistical values

### Synopsis

```
<operator> ifile ofile
```

### Description

This module computes statistical values over time steps of the same day. Depending on the actual operator the minimum, maximum, sum, average, variance or standard deviation of time steps of the same day is written to **ofile**. The date information for a time step in **ofile** is the date of the last contributing time step in **ifile**.

### Operators

<b>daymin</b>	Daily minimum For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same day, it is: $o(t, x) = \mathbf{min}\{i(t', x), t_1 < t' \leq t_n\}$
<b>daymax</b>	Daily maximum For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same day, it is: $o(t, x) = \mathbf{max}\{i(t', x), t_1 < t' \leq t_n\}$
<b>daysum</b>	Daily sum For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same day, it is: $o(t, x) = \mathbf{sum}\{i(t', x), t_1 < t' \leq t_n\}$
<b>daymean</b>	Daily mean For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same day, it is: $o(t, x) = \mathbf{mean}\{i(t', x), t_1 < t' \leq t_n\}$
<b>dayavg</b>	Daily average For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same day, it is: $o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$
<b>dayvar</b>	Daily variance For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same day, it is: $o(t, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$
<b>daystd</b>	Daily standard deviation For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same day, it is: $o(t, x) = \mathbf{std}\{i(t', x), t_1 < t' \leq t_n\}$

### Example

To compute the daily mean of a time series, use:

```
cdo daymean ifile ofile
```

## 2.8.15. DAYPCTL - Daily percentile values

### Synopsis

```
daypctl,p ifile1 ifile2 ifile3 ofile
```

### Description

This module computes percentiles over all time steps in **ifile1** of the same day. The algorithm uses histograms with minimum and maximum bounds given in **ifile2** and **ifile3**, respectively. The default number of histogram bins is 100. The default can be overridden by defining the environment variable **CDO\_PCTL\_NBINS**. The files **ifile2** and **ifile3** must be the result of corresponding **daymin** and **daymax** operations, respectively. The date information for a time step in **ofile** is the date of the last contributing time step in **ifile1**.

For every adjacent sequence  $t_1, \dots, t_n$  of time steps of the same day, it is:

$$o(t, x) = \text{pth percentile}\{i(t', x), t_1 < t' \leq t_n\}$$

### Parameter

*p*     INTEGER     Percentile number in 1, ..., 99

### Environment

**CDO\_PCTL\_NBINS**     Sets the number of histogram bins. The default number is 100.

### Example

To compute the daily 90th percentile of a time series, use:

```
cdo daymin ifile minfile
cdo daymax ifile maxfile
cdo daypctl,90 ifile minfile maxfile ofile
```

Or shorter using operator piping:

```
cdo daypctl,90 ifile -daymin ifile -daymax ifile ofile
```

## 2.8.16. MONSTAT - Monthly statistical values

### Synopsis

```
<operator> ifile ofile
```

### Description

This module computes statistical values over time steps of the same month. Depending on the actual operator the minimum, maximum, sum, average, variance or standard deviation of time steps of the same month is written to `ofile`. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

### Operators

<b>monmin</b>	Monthly minimum For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same month, it is: $o(t, x) = \mathbf{min}\{i(t', x), t_1 < t' \leq t_n\}$
<b>monmax</b>	Monthly maximum For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same month, it is: $o(t, x) = \mathbf{max}\{i(t', x), t_1 < t' \leq t_n\}$
<b>monsum</b>	Monthly sum For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same month, it is: $o(t, x) = \mathbf{sum}\{i(t', x), t_1 < t' \leq t_n\}$
<b>monmean</b>	Monthly mean For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same month, it is: $o(t, x) = \mathbf{mean}\{i(t', x), t_1 < t' \leq t_n\}$
<b>monavg</b>	Monthly average For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same month, it is: $o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$
<b>monvar</b>	Monthly variance For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same month, it is: $o(t, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$
<b>monstd</b>	Monthly standard deviation For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same month, it is: $o(t, x) = \mathbf{std}\{i(t', x), t_1 < t' \leq t_n\}$

### Example

To compute the monthly mean of a time series, use:

```
cdo monmean ifile ofile
```

## 2.8.17. MONPCTL - Monthly percentile values

### Synopsis

```
monpctl,p ifile1 ifile2 ifile3 ofile
```

### Description

This module computes percentiles over all time steps in *ifile1* of the same month. The algorithm uses histograms with minimum and maximum bounds given in *ifile2* and *ifile3*, respectively. The default number of histogram bins is 100. The default can be overridden by setting the environment variable CDO\_PCTL\_NBINS to a different value. The files *ifile2* and *ifile3* must be the result of corresponding monmin and monmax operations, respectively. The date information for a time step in *ofile* is the date of the last contributing time step in *ifile1*.

For every adjacent sequence  $t_1, \dots, t_n$  of time steps of the same month, it is:

$$o(t, x) = \text{pth percentile}\{i(t', x), t_1 < t' \leq t_n\}$$

### Parameter

*p*     INTEGER     Percentile number in 1, ..., 99

### Environment

CDO\_PCTL\_NBINS     Sets the number of histogram bins. The default number is 100.

### Example

To compute the monthly 90th percentile of a time series, use:

```
cdo monmin ifile minfile
cdo monmax ifile maxfile
cdo monpctl,90 ifile minfile maxfile ofile
```

Or shorter using operator piping:

```
cdo monpctl,90 ifile -monmin ifile -monmax ifile ofile
```

## 2.8.18. YEARSTAT - Yearly statistical values

### Synopsis

```
<operator> ifile ofile
```

### Description

This module computes statistical values over time steps of the same year. Depending on the actual operator the minimum, maximum, sum, average, variance or standard deviation of time steps of the same year is written to **ofile**. The date information for a time step in **ofile** is the date of the last contributing time step in **ifile**.

### Operators

<b>yearmin</b>	Yearly minimum For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same year, it is: $o(t, x) = \mathbf{min}\{i(t', x), t_1 < t' \leq t_n\}$
<b>yearmax</b>	Yearly maximum For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same year, it is: $o(t, x) = \mathbf{max}\{i(t', x), t_1 < t' \leq t_n\}$
<b>yearsum</b>	Yearly sum For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same year, it is: $o(t, x) = \mathbf{sum}\{i(t', x), t_1 < t' \leq t_n\}$
<b>yearmean</b>	Yearly mean For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same year, it is: $o(t, x) = \mathbf{mean}\{i(t', x), t_1 < t' \leq t_n\}$
<b>yearavg</b>	Yearly average For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same year, it is: $o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$
<b>yearvar</b>	Yearly variance For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same year, it is: $o(t, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$
<b>yearstd</b>	Yearly standard deviation For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same year, it is: $o(t, x) = \mathbf{std}\{i(t', x), t_1 < t' \leq t_n\}$

### Note

The operators **yearmean** and **yearavg** computes only an arithmetical mean!

### Example

To compute the yearly mean of a time series, use:

```
cdo yearmean ifile ofile
```

To compute the yearly mean from the correct weighted monthly mean, use:

```
cdo mulc,12 -divdpy -yearmean -muldpm ifile ofile
```



## 2.8.19. YEARPCTL - Yearly percentile values

### Synopsis

```
yearpctl,p ifile1 ifile2 ifile3 ofile
```

### Description

This module computes percentiles over all time steps in `ifile1` of the same year. The algorithm uses histograms with minimum and maximum bounds given in `ifile2` and `ifile3`, respectively. The default number of histogram bins is 100. The default can be overridden by setting the environment variable `CDO_PCTL_NBINS` to a different value. The files `ifile2` and `ifile3` must be the result of corresponding `yearmin` and `yearmax` operations, respectively. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

For every adjacent sequence  $t_1, \dots, t_n$  of time steps of the same year, it is:

$$o(t, x) = \text{pth percentile}\{i(t', x), t_1 < t' \leq t_n\}$$

### Parameter

`p`     INTEGER     Percentile number in 1, ..., 99

### Environment

`CDO_PCTL_NBINS`     Sets the number of histogram bins. The default number is 100.

### Example

To compute the yearly 90th percentile of a time series, use:

```
cdo yearmin ifile minfile
cdo yearmax ifile maxfile
cdo yearpctl,90 ifile minfile maxfile ofile
```

Or shorter using operator piping:

```
cdo yearpctl,90 ifile -yearmin ifile -yearmax ifile ofile
```

## 2.8.20. SEASSTAT - Seasonal statistical values

### Synopsis

```
<operator> ifile ofile
```

### Description

This module computes statistical values over time steps of the same season. Depending on the actual operator the minimum, maximum, sum, average, variance or standard deviation of time steps of the same season is written to `ofile`. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`. Be careful about the first and the last output time step, they may be incorrect values if the seasons have incomplete time steps.

### Operators

<b>seasmin</b>	Seasonal minimum For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same season, it is: $o(t, x) = \mathbf{min}\{i(t', x), t_1 < t' \leq t_n\}$
<b>seasmax</b>	Seasonal maximum For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same season, it is: $o(t, x) = \mathbf{max}\{i(t', x), t_1 < t' \leq t_n\}$
<b>seassum</b>	Seasonal sum For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same season, it is: $o(t, x) = \mathbf{sum}\{i(t', x), t_1 < t' \leq t_n\}$
<b>seasmean</b>	Seasonal mean For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same season, it is: $o(t, x) = \mathbf{mean}\{i(t', x), t_1 < t' \leq t_n\}$
<b>seasavg</b>	Seasonal average For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same season, it is: $o(t, x) = \mathbf{avg}\{i(t', x), t_1 < t' \leq t_n\}$
<b>seasvar</b>	Seasonal variance For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same season, it is: $o(t, x) = \mathbf{var}\{i(t', x), t_1 < t' \leq t_n\}$
<b>seasstd</b>	Seasonal standard deviation For every adjacent sequence $t_1, \dots, t_n$ of time steps of the same season, it is: $o(t, x) = \mathbf{std}\{i(t', x), t_1 < t' \leq t_n\}$

### Example

To compute the seasonal mean of a time series, use:

```
cdo seasmean ifile ofile
```

## 2.8.21. SEASPCTL - Seasonal percentile values

### Synopsis

```
seaspctl,p ifile1 ifile2 ifile3 ofile
```

### Description

This module computes percentiles over all time steps in `ifile1` of the same season. The algorithm uses histograms with minimum and maximum bounds given in `ifile2` and `ifile3`, respectively. The default number of histogram bins is 100. The default can be overridden by setting the environment variable `CDO_PCTL_NBINS` to a different value. The files `ifile2` and `ifile3` must be the result of corresponding `seasmin` and `seasmax` operations, respectively. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`. Be careful about the first and the last output time step, they may be incorrect values if the seasons have incomplete time steps.

For every adjacent sequence  $t_1, \dots, t_n$  of time steps of the same season, it is:

$$o(t, x) = \text{pth percentile}\{i(t', x), t_1 < t' \leq t_n\}$$

### Parameter

`p`     INTEGER     Percentile number in 1, ..., 99

### Environment

`CDO_PCTL_NBINS`     Sets the number of histogram bins. The default number is 100.

### Example

To compute the seasonal 90th percentile of a time series, use:

```
cdo seasmin ifile minfile
cdo seasmax ifile maxfile
cdo seaspctl,90 ifile minfile maxfile ofile
```

Or shorter using operator piping:

```
cdo seaspctl,90 ifile -seasmin ifile -seasmax ifile ofile
```

## 2.8.22. YHOURSTAT - Multi-year hourly statistical values

### Synopsis

`<operator> ifile ofile`

### Description

This module writes to `ofile`, according to the actual operator, the minimum, maximum, sum, average, variance or standard deviation of each hour and day of year in `ifile`. The date information in an output field is the date of the last contributing input field.

### Operators

<b>yhourmin</b>	Multi-year hourly minimum $o(0001, x) = \mathbf{min}\{i(t, x), \text{day}(i(t)) = 0001\}$ $\vdots$ $o(8784, x) = \mathbf{min}\{i(t, x), \text{day}(i(t)) = 8784\}$
<b>yhourmax</b>	Multi-year hourly maximum $o(0001, x) = \mathbf{max}\{i(t, x), \text{day}(i(t)) = 0001\}$ $\vdots$ $o(8784, x) = \mathbf{max}\{i(t, x), \text{day}(i(t)) = 8784\}$
<b>yhoursum</b>	Multi-year hourly sum $o(0001, x) = \mathbf{sum}\{i(t, x), \text{day}(i(t)) = 0001\}$ $\vdots$ $o(8784, x) = \mathbf{sum}\{i(t, x), \text{day}(i(t)) = 8784\}$
<b>yhourmean</b>	Multi-year hourly mean $o(0001, x) = \mathbf{mean}\{i(t, x), \text{day}(i(t)) = 0001\}$ $\vdots$ $o(8784, x) = \mathbf{mean}\{i(t, x), \text{day}(i(t)) = 8784\}$
<b>yhouravg</b>	Multi-year hourly average $o(0001, x) = \mathbf{avg}\{i(t, x), \text{day}(i(t)) = 0001\}$ $\vdots$ $o(8784, x) = \mathbf{avg}\{i(t, x), \text{day}(i(t)) = 8784\}$
<b>yhourvar</b>	Multi-year hourly variance $o(0001, x) = \mathbf{var}\{i(t, x), \text{day}(i(t)) = 0001\}$ $\vdots$ $o(8784, x) = \mathbf{var}\{i(t, x), \text{day}(i(t)) = 8784\}$
<b>yhourstd</b>	Multi-year hourly standard deviation $o(0001, x) = \mathbf{std}\{i(t, x), \text{day}(i(t)) = 0001\}$ $\vdots$ $o(8784, x) = \mathbf{std}\{i(t, x), \text{day}(i(t)) = 8784\}$

### Example

To compute the hourly mean for all days over all input years, use:

```
cdo yhourmean ifile ofile
```

## 2.8.23. YDAYSTAT - Multi-year daily statistical values

### Synopsis

`<operator> ifile ofile`

### Description

This module writes to `ofile`, according to the actual operator, the minimum, maximum, sum, average, variance or standard deviation of each day of year in `ifile`. The date information in an output field is the date of the last contributing input field.

### Operators

<b>ydaymin</b>	Multi-year daily minimum $o(001, x) = \min\{i(t, x), \text{day}(i(t)) = 001\}$ $\vdots$ $o(366, x) = \min\{i(t, x), \text{day}(i(t)) = 366\}$
<b>ydaymax</b>	Multi-year daily maximum $o(001, x) = \max\{i(t, x), \text{day}(i(t)) = 001\}$ $\vdots$ $o(366, x) = \max\{i(t, x), \text{day}(i(t)) = 366\}$
<b>ydaysum</b>	Multi-year daily sum $o(001, x) = \text{sum}\{i(t, x), \text{day}(i(t)) = 001\}$ $\vdots$ $o(366, x) = \text{sum}\{i(t, x), \text{day}(i(t)) = 366\}$
<b>ydaymean</b>	Multi-year daily mean $o(001, x) = \text{mean}\{i(t, x), \text{day}(i(t)) = 001\}$ $\vdots$ $o(366, x) = \text{mean}\{i(t, x), \text{day}(i(t)) = 366\}$
<b>ydayavg</b>	Multi-year daily average $o(001, x) = \text{avg}\{i(t, x), \text{day}(i(t)) = 001\}$ $\vdots$ $o(366, x) = \text{avg}\{i(t, x), \text{day}(i(t)) = 366\}$
<b>ydayvar</b>	Multi-year daily variance $o(001, x) = \text{var}\{i(t, x), \text{day}(i(t)) = 001\}$ $\vdots$ $o(366, x) = \text{var}\{i(t, x), \text{day}(i(t)) = 366\}$
<b>ydaystd</b>	Multi-year daily standard deviation $o(001, x) = \text{std}\{i(t, x), \text{day}(i(t)) = 001\}$ $\vdots$ $o(366, x) = \text{std}\{i(t, x), \text{day}(i(t)) = 366\}$

### Example

To compute the daily mean over all input years, use:

```
cdo ydaymean ifile ofile
```

## 2.8.24. YDAYPCTL - Multi-year daily percentile values

### Synopsis

```
ydaypctl,p ifile1 ifile2 ifile3 ofile
```

### Description

This module writes to `ofile` a certain percentile of each day of year in `ifile1`. The algorithm uses histograms with minimum and maximum bounds given in `ifile2` and `ifile3`, respectively. The default number of histogram bins is 100. The default can be overridden by setting the environment variable `CDO_PCTL_NBINS` to a different value. The files `ifile2` and `ifile3` must 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.

$$o(001, x) = \text{pth percentile}\{i(t, x), \text{day}(i(t)) = 001\}$$

$$\vdots$$

$$o(366, x) = \text{pth percentile}\{i(t, x), \text{day}(i(t)) = 366\}$$

### Parameter

`p`     INTEGER     Percentile number in 1, ..., 99

### Environment

`CDO_PCTL_NBINS`     Sets the number of histogram bins. The default number is 100.

### Example

To compute the daily 90th percentile over all input years, use:

```
cdo ydaymin ifile minfile
cdo ydaymax ifile maxfile
cdo ydaypctl,90 ifile minfile maxfile ofile
```

Or shorter using operator piping:

```
cdo ydaypctl,90 ifile -ydaymin ifile -ydaymax ifile ofile
```

## 2.8.25. YMONSTAT - Multi-year monthly statistical values

### Synopsis

`<operator> ifile ofile`

### Description

This module writes to `ofile`, according to the actual operator, the minimum, maximum, sum, average, variance or standard deviation of each month of year in `ifile`. The date information in an output field is the date of the last contributing input field.

### Operators

<b>ymin</b>	Multi-year monthly minimum $o(01, x) = \min\{i(t, x), \text{month}(i(t)) = 01\}$ $\vdots$ $o(12, x) = \min\{i(t, x), \text{month}(i(t)) = 12\}$
<b>ymax</b>	Multi-year monthly maximum $o(01, x) = \max\{i(t, x), \text{month}(i(t)) = 01\}$ $\vdots$ $o(12, x) = \max\{i(t, x), \text{month}(i(t)) = 12\}$
<b>ysum</b>	Multi-year monthly sum $o(01, x) = \text{sum}\{i(t, x), \text{month}(i(t)) = 01\}$ $\vdots$ $o(12, x) = \text{sum}\{i(t, x), \text{month}(i(t)) = 12\}$
<b>ymean</b>	Multi-year monthly mean $o(01, x) = \text{mean}\{i(t, x), \text{month}(i(t)) = 01\}$ $\vdots$ $o(12, x) = \text{mean}\{i(t, x), \text{month}(i(t)) = 12\}$
<b>yavg</b>	Multi-year monthly average $o(01, x) = \text{avg}\{i(t, x), \text{month}(i(t)) = 01\}$ $\vdots$ $o(12, x) = \text{avg}\{i(t, x), \text{month}(i(t)) = 12\}$
<b>yvar</b>	Multi-year monthly variance $o(01, x) = \text{var}\{i(t, x), \text{month}(i(t)) = 01\}$ $\vdots$ $o(12, x) = \text{var}\{i(t, x), \text{month}(i(t)) = 12\}$
<b>ystd</b>	Multi-year monthly standard deviation $o(01, x) = \text{std}\{i(t, x), \text{month}(i(t)) = 01\}$ $\vdots$ $o(12, x) = \text{std}\{i(t, x), \text{month}(i(t)) = 12\}$

### Example

To compute the monthly mean over all input years, use:

```
cdo ymonmean ifile ofile
```

## 2.8.26. YMONPCTL - Multi-year monthly percentile values

### Synopsis

```
ymonpctl,p ifile1 ifile2 ifile3 ofile
```

### Description

This module writes to `ofile` a certain percentile of each month of year in `ifile1`. The algorithm uses histograms with minimum and maximum bounds given in `ifile2` and `ifile3`, respectively. The default number of histogram bins is 100. The default can be overridden by setting the environment variable `CDO_PCTL_NBINS` to a different value. The files `ifile2` and `ifile3` must 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.

$$o(01, x) = \text{pth percentile}\{i(t, x), \text{month}(i(t)) = 01\}$$

$$\vdots$$

$$o(12, x) = \text{pth percentile}\{i(t, x), \text{month}(i(t)) = 12\}$$

### Parameter

*p*     INTEGER     Percentile number in 1, ..., 99

### Environment

`CDO_PCTL_NBINS`     Sets the number of histogram bins. The default number is 100.

### Example

To compute the monthly 90th percentile over all input years, use:

```
cdo ymonmin ifile minfile
cdo ymonmax ifile maxfile
cdo ymonpctl,90 ifile minfile maxfile ofile
```

Or shorter using operator piping:

```
cdo ymonpctl,90 ifile -ymonmin ifile -ymonmax ifile ofile
```



## 2.8.27. YSEASSTAT - Multi-year seasonal statistical values

### Synopsis

`<operator> ifile ofile`

### Description

This module writes to `ofile`, according to the actual operator, the minimum, maximum, sum, average, variance or standard deviation of each season in `ifile`. The date information in an output field is the date of the last contributing input field.

### Operators

<b>yseasmin</b>	Multi-year seasonal minimum $o(1, x) = \min\{i(t, x), \text{month}(i(t)) = 12, 01, 02\}$ $o(2, x) = \min\{i(t, x), \text{month}(i(t)) = 03, 04, 05\}$ $o(3, x) = \min\{i(t, x), \text{month}(i(t)) = 06, 07, 08\}$ $o(4, x) = \min\{i(t, x), \text{month}(i(t)) = 09, 10, 11\}$
<b>yseasmax</b>	Multi-year seasonal maximum $o(1, x) = \max\{i(t, x), \text{month}(i(t)) = 12, 01, 02\}$ $o(2, x) = \max\{i(t, x), \text{month}(i(t)) = 03, 04, 05\}$ $o(3, x) = \max\{i(t, x), \text{month}(i(t)) = 06, 07, 08\}$ $o(4, x) = \max\{i(t, x), \text{month}(i(t)) = 09, 10, 11\}$
<b>yseassum</b>	Multi-year seasonal sum $o(1, x) = \text{sum}\{i(t, x), \text{month}(i(t)) = 12, 01, 02\}$ $o(2, x) = \text{sum}\{i(t, x), \text{month}(i(t)) = 03, 04, 05\}$ $o(3, x) = \text{sum}\{i(t, x), \text{month}(i(t)) = 06, 07, 08\}$ $o(4, x) = \text{sum}\{i(t, x), \text{month}(i(t)) = 09, 10, 11\}$
<b>yseasmean</b>	Multi-year seasonal mean $o(1, x) = \text{mean}\{i(t, x), \text{month}(i(t)) = 12, 01, 02\}$ $o(2, x) = \text{mean}\{i(t, x), \text{month}(i(t)) = 03, 04, 05\}$ $o(3, x) = \text{mean}\{i(t, x), \text{month}(i(t)) = 06, 07, 08\}$ $o(4, x) = \text{mean}\{i(t, x), \text{month}(i(t)) = 09, 10, 11\}$
<b>yseasavg</b>	Multi-year seasonal average $o(1, x) = \text{avg}\{i(t, x), \text{month}(i(t)) = 12, 01, 02\}$ $o(2, x) = \text{avg}\{i(t, x), \text{month}(i(t)) = 03, 04, 05\}$ $o(3, x) = \text{avg}\{i(t, x), \text{month}(i(t)) = 06, 07, 08\}$ $o(4, x) = \text{avg}\{i(t, x), \text{month}(i(t)) = 09, 10, 11\}$
<b>yseasvar</b>	Multi-year seasonal variance $o(1, x) = \text{var}\{i(t, x), \text{month}(i(t)) = 12, 01, 02\}$ $o(2, x) = \text{var}\{i(t, x), \text{month}(i(t)) = 03, 04, 05\}$ $o(3, x) = \text{var}\{i(t, x), \text{month}(i(t)) = 06, 07, 08\}$ $o(4, x) = \text{var}\{i(t, x), \text{month}(i(t)) = 09, 10, 11\}$
<b>yseasstd</b>	Multi-year seasonal standard deviation $o(1, x) = \text{std}\{i(t, x), \text{month}(i(t)) = 12, 01, 02\}$ $o(2, x) = \text{std}\{i(t, x), \text{month}(i(t)) = 03, 04, 05\}$ $o(3, x) = \text{std}\{i(t, x), \text{month}(i(t)) = 06, 07, 08\}$ $o(4, x) = \text{std}\{i(t, x), \text{month}(i(t)) = 09, 10, 11\}$

## Example

To compute the seasonal mean over all input years, use:

```
cdo yseasmean ifile ofile
```

## 2.8.28. YSEASPCTL - Multi-year seasonal percentile values

### Synopsis

```
yseaspctl,p ifile1 ifile2 ifile3 ofile
```

### Description

This module writes to `ofile` a certain percentile of each season in `ifile1`. The algorithm uses histograms with minimum and maximum bounds given in `ifile2` and `ifile3`, respectively. The default number of histogram bins is 100. The default can be overridden by setting the environment variable `CDO_PCTL_NBINS` to a different value. The files `ifile2` and `ifile3` must 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.

$$o(1, x) = \text{pth percentile}\{i(t, x), \text{month}(i(t)) = 12, 01, 02\}$$

$$o(2, x) = \text{pth percentile}\{i(t, x), \text{month}(i(t)) = 03, 04, 05\}$$

$$o(3, x) = \text{pth percentile}\{i(t, x), \text{month}(i(t)) = 06, 07, 08\}$$

$$o(4, x) = \text{pth percentile}\{i(t, x), \text{month}(i(t)) = 09, 10, 11\}$$

### Parameter

`p`     INTEGER     Percentile number in 1, ..., 99

### Environment

`CDO_PCTL_NBINS`     Sets the number of histogram bins. The default number is 100.

## Example

To compute the seasonal 90th percentile over all input years, use:

```
cdo yseasmin ifile minfile
cdo yseasmax ifile maxfile
cdo yseaspctl,90 ifile minfile maxfile ofile
```

Or shorter using operator piping:

```
cdo yseaspctl,90 ifile -yseasmin ifile -yseasmax ifile ofile
```

## 2.8.29. YDRUNSTAT - Multi-year daily running statistical values

### Synopsis

`<operator>,nts ifile ofile`

### Description

This module writes to `ofile` running statistical values for each day of year in `ifile`. Depending on the actual operator, the minimum, maximum, sum, average, variance or standard deviation of all time steps in running windows of which the medium time step corresponds to a certain day of year is computed. The date information in an output field is the date of the medium time step in the last contributing running window. Note that the operator must 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  $(nts-1)/2$  time steps after the first time step of the input time series and ends  $(nts-1)/2$  time steps 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 yields physically meaningful results only if the input time series does include the  $(nts-1)/2$  days before and after each period of interest.

### Operators

<b>ydrunmin</b>	Multi-year daily running minimum $o(001, x) = \min\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 001\}$ $\vdots$ $o(366, x) = \min\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 366\}$
<b>ydrunmax</b>	Multi-year daily running maximum $o(001, x) = \max\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 001\}$ $\vdots$ $o(366, x) = \max\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 366\}$
<b>ydrunsum</b>	Multi-year daily running sum $o(001, x) = \sum\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 001\}$ $\vdots$ $o(366, x) = \sum\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 366\}$
<b>ydrunmean</b>	Multi-year daily running mean $o(001, x) = \text{mean}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 001\}$ $\vdots$ $o(366, x) = \text{mean}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 366\}$
<b>ydrunavg</b>	Multi-year daily running average $o(001, x) = \text{avg}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 001\}$ $\vdots$ $o(366, x) = \text{avg}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 366\}$
<b>ydrunvar</b>	Multi-year daily running variance $o(001, x) = \text{var}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 001\}$ $\vdots$ $o(366, x) = \text{var}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 366\}$
<b>ydrunstd</b>	Multi-year daily running standard deviation $o(001, x) = \text{std}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 001\}$ $\vdots$ $o(366, x) = \text{std}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 366\}$

## Parameter

*nts*    INTEGER    Number of time steps

## Example

Assume the input data provide a continuous time series of daily measurements. To compute the running multi-year daily mean over all input time steps for a running window of five days, use:

```
cdo ydrunmean,5 ifile ofile
```

Note that except for the standard deviation the results of the operators in this module are equivalent to a composition of corresponding operators from the [YDAYSTAT](#) and [RUNSTAT](#) modules. For instance, the above command yields the same result as:

```
cdo ydaymean -runmean,5 ifile ofile
```

## 2.8.30. YDRUNPCTL - Multi-year daily running percentile values

### Synopsis

```
ydrunctl,p,nts ifile1 ifile2 ifile3 ofile
```

### Description

This module writes to `ofile` running percentile values for each day of year in `ifile1`. A certain percentile is computed for all time steps in running windows of which the medium time step corresponds to a certain day of year. The algorithm uses histograms with minimum and maximum bounds given in `ifile2` and `ifile3`, respectively. The default number of histogram bins is 100. The default can be overridden by setting the environment variable `CDO_PCTL_NBINS` to a different value. The files `ifile2` and `ifile3` must be the result of corresponding `ydrunmin` and `ydrunmax` operations, respectively. The date information in an output field is the date of the medium time step in the last contributing running window. Note that the operator must 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  $(nts-1)/2$  time steps after the first time step of the input time series and ends  $(nts-1)/2$  time steps 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 yields physically meaningful results only if the input time series does include the  $(nts-1)/2$  days before and after each period of interest.

$$o(001, x) = \text{pth percentile}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 001\}$$

$$\vdots$$

$$o(366, x) = \text{pth percentile}\{i(t, x), i(t+1, x), \dots, i(t+nts-1, x); \text{day}[(i(t+(nts-1)/2)] = 366\}$$

### Parameter

<code>p</code>	INTEGER	Percentile number in 1, ..., 99
<code>nts</code>	INTEGER	Number of time steps

### Environment

<code>CDO_PCTL_NBINS</code>	Sets the number of histogram bins. The default number is 100.
-----------------------------	---

### Example

Assume the input data provide a continuous time series of daily measurements. To compute the running multi-year daily 90th percentile over all input time steps for a running window of five days, use:

```
cdo ydrunmin,5 ifile minfile
cdo ydrunmax,5 ifile maxfile
cdo ydrunctl,90,5 ifile minfile maxfile ofile
```

Or shorter using operator piping:

```
cdo ydrunctl,90,5 ifile -ydrunmin ifile -ydrunmax ifile ofile
```

## 2.9. Regression

This sections contains modules for linear regression of time series.

Here is a short overview of all operators in this section:

<b>regres</b>	Regression
<b>detrend</b>	Detrend
<b>trend</b>	Trend
<b>subtrend</b>	Subtract trend

### 2.9.1. REGRES - Regression

#### Synopsis

```
regres ifile ofile
```

#### Description

The values of the input file `ifile` are assumed to be distributed as  $N(a + bt, \sigma^2)$  with unknown  $a$ ,  $b$  and  $\sigma^2$ . This operator estimates the parameter  $b$ . For every field element  $x$  only those time steps  $t$  belong to the sample  $S(x)$ , which have  $i(t, x) \neq \text{miss}$ . It is

$$o(1, x) = \frac{\sum_{t \in S(x)} \left( i(t, x) - \frac{1}{\#S(x)} \sum_{t' \in S(x)} i(t', x) \right) \left( t - \frac{1}{\#S(x)} \sum_{t' \in S(x)} t' \right)}{\sum_{t \in S(x)} \left( t - \frac{1}{\#S(x)} \sum_{t' \in S(x)} t' \right)^2}$$

### 2.9.2. DETREND - Detrend time series

#### Synopsis

```
detrend ifile ofile
```

#### Description

Every time series in `ifile` is linearly detrended. For every field element  $x$  only those time steps  $t$  belong to the sample  $S(x)$ , which have  $i(t, x) \neq \text{miss}$ . With

$$a(x) = \frac{1}{\#S(x)} \sum_{t \in S(x)} i(t, x) - b(x) \left( \frac{1}{\#S(x)} \sum_{t \in S(x)} t \right)$$

and

$$b(x) = \frac{\sum_{t \in S(x)} \left( i(t, x) - \frac{1}{\#S(x)} \sum_{t' \in S(x)} i(t', x) \right) \left( t - \frac{1}{\#S(x)} \sum_{t' \in S(x)} t' \right)}{\sum_{t \in S(x)} \left( t - \frac{1}{\#S(x)} \sum_{t' \in S(x)} t' \right)^2}$$

it is

$$o(t, x) = i(t, x) - (a(x) + b(x)t)$$

#### Note

This operator has to keep the fields of all time steps concurrently in the memory. If not enough memory is available, use the operators [trend](#) and [subtrend](#).

#### Example

To detrend the data in `ifile` and to store the detrended data in `ofile`, use:

```
cdo detrend ifile ofile
```

### 2.9.3. TREND - Trend of time series

#### Synopsis

```
trend ifile ofile1 ofile2
```

#### Description

The values of the input file `ifile` are assumed to be distributed as  $N(a + bt, \sigma^2)$  with unknown  $a$ ,  $b$  and  $\sigma^2$ . This operator estimates the parameter  $a$  and  $b$ . For every field element  $x$  only those time steps  $t$  belong to the sample  $S(x)$ , which have  $i(t, x) \neq \text{miss}$ . It is

$$o_1(1, x) = \frac{1}{\#S(x)} \sum_{t \in S(x)} i(t, x) - b(x) \left( \frac{1}{\#S(x)} \sum_{t \in S(x)} t \right)$$

and

$$o_2(1, x) = \frac{\sum_{t \in S(x)} \left( i(t, x) - \frac{1}{\#S(x)} \sum_{t' \in S(x)} i(t', x) \right) \left( t - \frac{1}{\#S(x)} \sum_{t' \in S(x)} t' \right)}{\sum_{t \in S(x)} \left( t - \frac{1}{\#S(x)} \sum_{t' \in S(x)} t' \right)^2}$$

Thus the estimation for  $a$  is stored in `ofile1` and that for  $b$  is stored in `ofile2`. To subtract the trend from the data see operator [subtrend](#).

### 2.9.4. SUBTREND - Subtract a trend

#### Synopsis

```
subtrend ifile1 ifile2 ifile3 ofile
```

#### Description

This operator is for subtracting a trend computed by the operator [trend](#). It is

$$o(t, x) = i_1(t, x) - (i_2(1, x) + i_3(1, x) \cdot t)$$

where  $t$  is the time steps.

#### Example

The typical call for detrend the data in `ifile` and to store the detrended data in `ofile` is:

```
cdo trend ifile afile bfile
cdo subtrend ifile afile bfile ofile
```

The result is identical to operator [detrend](#):

```
cdo detrend ifile ofile
```



## 2.10. Interpolation

This section contains modules to interpolate datasets. There are several operators to interpolate horizontal fields to a new grid. Some of those operators can handle only 2D fields on a regular rectangular grid. Vertical interpolation of 3D variables is possible from hybrid model level to height or pressure level. Interpolation in time is possible between time steps and between years.

Here is a short overview of all operators in this section:

<b>remapbil</b>	Bilinear interpolation
<b>remapbic</b>	Bicubic interpolation
<b>remapcon</b>	Conservative remapping
<b>remapdis</b>	Distance-weighted averaging
<b>genbil</b>	Generate bilinear interpolation weights
<b>genbic</b>	Generate bicubic interpolation weights
<b>gencon</b>	Generate conservative interpolation weights
<b>gendis</b>	Generate distance-weighted averaging weights
<b>remap</b>	SCRIP grid remapping
<b>interpolate</b>	PINGO grid interpolation
<b>intgridbil</b>	Bilinear grid interpolation
<b>remapeta</b>	Remap vertical hybrid level
<b>ml2pl</b>	Model to pressure level interpolation
<b>ml2hl</b>	Model to height level interpolation
<b>inttime</b>	Time interpolation
<b>intntime</b>	Time interpolation
<b>intyear</b>	Year interpolation

## 2.10.1. REMAPGRID - SCRIP grid interpolation

### Synopsis

```
<operator>,grid ifile ofile
```

### Description

This module contains operators to interpolate all input fields to a new grid. Each operator is using a different remapping method. The interpolation is based on a special SCRIP library version. For a detailed description of the remapping methods see [\[SCRIP\]](#).

### Operators

<b>remapbil</b>	Bilinear interpolation Performs a bilinear interpolation on all input fields. This interpolation method works only on rectangular grids.
<b>remapbic</b>	Bicubic interpolation Performs a bicubic interpolation on all input fields. This interpolation method works only on rectangular grids.
<b>remapcon</b>	Conservative remapping Performs a first order conservative remapping on all input fields.
<b>remapdis</b>	Distance-weighted averaging Performs a distance-weighted average of the four nearest neighbor values on all input fields.

### Parameter

<i>grid</i>	STRING	Target grid description file or name
-------------	--------	--------------------------------------

### Environment

NORMALIZE_OPT	This variable is used to choose the normalization of the conservative remapping. By default, NORMALIZE_OPT is set to be 'fracarea' and will include the destination area fraction in the output weights; other options are 'none' and 'destarea' (for more information see <a href="#">[SCRIP]</a> ).
---------------	---

### Note

For this program the author has converted the original Fortran 90 SCRIP software to ANSI C. In case of any problems send a bug report to CDO and not to SCRIP!

### Example

Say *ifile* contains fields on a rectangular grid. Remap all fields bilinear to a T42 gaussian grid:

```
cdo remapbil,t42grid ifile ofile
```

## 2.10.2. GENWEIGHTS - Generate SCRIP grid interpolation weights

### Synopsis

```
<operator> .grid ifile ofile
```

### Description

Grid interpolation can be a very time consuming process. Especially if the data is on an unstructured or on a large grid. In this case the [SCRIP](#) interpolation process can be split into two parts. First generation of the interpolation weights, this is the most time consuming part. These interpolation weights can be reused for every remapping process. This method works only if all input fields are on the same grid and a possibly mask (missing values) does not change. This module contains operators to generate SCRIP interpolation weights of the first input field. Each operator is using a different interpolation method.

### Operators

<b>genbil</b>	Generate bilinear interpolation weights Generates bilinear interpolation weights and write the result to a file. This interpolation method works only on rectangular grids.
<b>genbic</b>	Generate bicubic interpolation weights Generates bicubic interpolation weights and write the result to a file. This interpolation method works only on rectangular grids.
<b>gencon</b>	Generate conservative interpolation weights Generates first order conservative interpolation weights and write the result to a file.
<b>gendis</b>	Generate distance-weighted averaging weights Generates distance-weighted average weights of the four nearest neighbor values and write the result to a file.

### Parameter

<i>grid</i>	STRING	Target grid description file or name
-------------	--------	--------------------------------------

### Environment

NORMALIZE_OPT	This variable is used to choose the normalization of the conservative interpolation. By default, NORMALIZE_OPT is set to be 'fracarea' and will include the destination area fraction in the output weights; other options are 'none' and 'destarea' (for more information see <a href="#">[SCRIP]</a> ).
---------------	---

### Note

For this program the author has converted the original Fortran 90 SCRIP software to ANSI C. In case of any problems send a bug report to CDO and not to SCRIP!

### Example

Say *ifile* contains fields on a rectangular grid. Remap all fields bilinear to a T42 gaussian grid:

```
cdo genbil,t42grid ifile remapweights.nc
cdo remap,t42grid,remapweights ifile ofile
```

### 2.10.3. REMAP - SCRIP grid remapping

#### Synopsis

```
remap,grid,weights ifile ofile
```

#### Description

This operator remaps all input fields to a new grid. The remap type and the interpolation weights of one grid are read from a netCDF file. The netCDF file with the weights must follow the [SCRIP](#) convention. Normaly these weights come from a previous call to module [GENWEIGHTS](#) or was created by the original SCRIP package.

#### Parameter

<i>grid</i>	STRING	Target grid description file or name
<i>weights</i>	STRING	Interpolation weights (SCRIP netCDF file)

#### Note

For this program the author has converted the original Fortran 90 SCRIP software to ANSI C. In case of any problems send a bug report to CDO and not to SCRIP!

#### Example

Say *ifile* contains fields on a regular grid. Remap all fields bilinear to a T42 gaussian grid:

```
cdo genbil,t42grid ifile remapweights.nc
cdo remap,t42grid,remapweights ifile ofile
```

## 2.10.4. INTGRID - Grid interpolation

### Synopsis

```
<operator>,grid ifile ofile
```

### Description

This module contains operators to interpolate all input fields to a new grid. All interpolation methods in this module work only on rectangular grids.

### Operators

**interpolate** PINGO grid interpolation  
This is the grid interpolation from PINGO. The basis of the interpolation is an underlying continuous field which is constructed in the following way. For two neighboured longitudes  $x_1$  and  $x_2$  and two neighboured latitudes  $y_1$  and  $y_2$  of the input grid every point at longitude  $x$  and latitude  $y$  with  $x_1 \leq x \leq x_2$  and  $y_1 \leq y \leq y_2$  is assigned the value

$$a = a_{11} + (a_{21} - a_{11}) \frac{x - x_1}{x_2 - x_1} + (a_{12} - a_{11}) \frac{y - y_1}{y_2 - y_1} + (a_{22} - a_{21} - a_{12} + a_{11}) \frac{(x - x_1)(y - y_1)}{(x_2 - x_1)(y_2 - y_1)}$$

where  $a_{ij}$  is the value at longitude  $x_i$  and latitude  $y_j$ . If one of the four values  $a_{11}$ ,  $a_{12}$ ,  $a_{21}$ ,  $a_{22}$  is the missing value, then  $a$  is also the missing value. Afterwards the underlying continuous field is expanded by a half mesh width. For a detailed description of this interpolation method see [\[PINGO\]](#).

**intgridbil** Bilinear grid interpolation  
Performs a bilinear interpolation on all input fields. This implementation is faster than [remapbil](#). The complete source grid must be inside the target grid. Missing values are not supported yet!

### Parameter

*grid*     STRING     Target grid description file or name

### Example

Say *ifile* contains fields on a rectangular grid. To interpolate all fields bilinear to a T42 gaussian grid, use:

```
cdo intgridbil,t42grid ifile ofile
```

## 2.10.5. REMAPETA - Remap vertical hybrid level

### Synopsis

```
remapeta,vct[,oro] ifile ofile
```

### 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

- first integration of hydrostatic equation
- extrapolation of surface pressure
- PBL profile interpolation
- interpolation in free atmosphere
- merging of both profiles
- 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 troposphere. Therefore the total mass above a given reference level is conserved. As reference level the geopotential height of the 500 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 profiles 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 [\[INTERA\]](#).

### Parameter

<i>vct</i>	STRING	File name of an ASCII dataset with the vertical coordinate table
<i>oro</i>	STRING	File name with the orography of the target dataset (optional)

### Example

To remap between different hybrid model level data, use:

```
cdo remapeta,vct ifile ofile
```

## 2.10.6. INTVERT - Vertical interpolation

### Synopsis

```
ml2pl,plevels ifile ofile
```

```
ml2hl,hlevels ifile ofile
```

### Description

Interpolate 3D variables on hybrid model level to pressure or height level. The input file must contain the log. surface pressure or the surface pressure. To interpolate the temperature, the orography is also needed. The pressure, temperature, and orography are identified by there code numbers. Supported code tables are: WMO standard table number 2 and ECMWF local table number 128. Use the alias ml2plx/ml2hlx or the environment variable EXTRAPOLATE to extrapolate missing values.

### Operators

**ml2pl**      Model to pressure level interpolation  
Interpolates 3D variables on hybrid model level to pressure level.

**ml2hl**      Model to height level interpolation  
Interpolates 3D variables on hybrid model level to height level. The procedure is the same as for operator [mh2pl](#) except that the pressure levels are calculated from the heights by:  
 $plevel = 101325 * \exp(hlevel / -7000)$

### Parameter

<i>plevels</i>	FLOAT	Pressure levels in pascal
<i>hlevels</i>	FLOAT	Height levels in meter (max level: 65535 m)

### Environment

EXTRAPOLATE	If set to 1 extrapolate missing values.
-------------	---

### Example

To interpolate hybrid model level data to pressure levels of 925, 850, 500 and 200 hPa, use:

```
cdo ml2pl,92500,85000,50000,20000 ifile ofile
```

## 2.10.7. INTTIME - Time interpolation

### Synopsis

```
inttime,date,time[,inc] ifile ofile
```

```
intntime,n ifile ofile
```

### Description

This module performs linear interpolation between time steps.

### Operators

<b>inttime</b>	Time interpolation This operator creates a new dataset by linear interpolation between time steps. The user has to define the start date/time with an optional increment.
<b>intntime</b>	Time interpolation This operator performs linear interpolation between time steps. The user has to define the number of time steps from one time step to the next.

### Parameter

<i>date</i>	STRING	Start date (format YYYY-MM-DD)
<i>time</i>	STRING	Start time (format hh:mm)
<i>inc</i>	STRING	Optional increment (minutes, hours, days, months, years) [default: 0hour]
<i>n</i>	INTEGER	Number of time steps from one time step to the next

### Example

Assumed a 6 hourly dataset starts at 1987-01-01 12:00. To interpolate this time series to a one hourly dataset, use:

```
cdo inttime,1987-01-01,12:00,1hour ifile ofile
```



## 2.10.8. INTYEAR - Year interpolation

### Synopsis

```
intyear,years ifile1 ifile2 oprefix
```

### Description

This operator performs linear interpolation between two years time step by time step. Appends four digits with the year to **oprefix** to form the output file names.

### Parameter

*years*      INTEGER      Comma separated list of years

### Example

Assumed you have two monthly mean datasets over a year. The first dataset has 12 time steps for year 1985 and the second for year 1990. To interpolate the years between 1985 and 1990 month by month, use:

```
cdo intyear,1986,1987,1988,1989 ifile1 ifile2 year
```

Example result of 'dir year\*' for netCDF datasets:

```
year1986.nc year1987.nc year1988.nc year1989.nc
```

## 2.11. Transformation

This section contains modules to perform spectral transformations.

Here is a short overview of all operators in this section:

<a href="#">sp2gp</a>	Spectral to gridpoint
<a href="#">sp2gpl</a>	Spectral to gridpoint (linear)
<a href="#">gp2sp</a>	Gridpoint to spectral
<a href="#">gp2spl</a>	Gridpoint to spectral (linear)
<a href="#">sp2sp</a>	Spectral to spectral
<a href="#">spcut</a>	Cut spectral wave number
<a href="#">dv2uv</a>	Divergence and vorticity to U and V wind
<a href="#">dv2uwl</a>	Divergence and vorticity to U and V wind (linear)
<a href="#">uv2dv</a>	U and V wind to divergence and vorticity
<a href="#">uv2dwl</a>	U and V wind to divergence and vorticity (linear)

### 2.11.1. SPECTRAL - Spectral transformation

#### Synopsis

```

sp2gp  ifile ofile
sp2gpl ifile ofile
gp2sp  ifile ofile
gp2spl ifile ofile
sp2sp, trunc ifile ofile
spcut, wnums ifile ofile

```

#### Description

This module transforms fields on gaussian grids to spectral coefficients and vice versa.

#### Operators

<b>sp2gp</b>	<p>Spectral to gridpoint</p> <p>Convert all fields with spectral coefficients to regular gaussian grid. The number of latitudes of the resulting gaussian grid is calculated from the triangular truncation by:</p> $nlat = NINT((trunc * \boxed{3} + 1.) / 2.)$
<b>sp2gpl</b>	<p>Spectral to gridpoint (linear)</p> <p>Convert all fields with spectral coefficients to regular gaussian grid. The number of latitudes of the resulting Gaussian grid is calculated from the triangular truncation by:</p> $nlat = NINT((trunc * \boxed{2} + 1.) / 2.)$ <p>Use this operator to convert ERA40 data e.g. from TL159 to N80.</p>
<b>gp2sp</b>	<p>Gridpoint to spectral</p> <p>Convert all gaussian gridpoint fields to spectral coefficients. The triangular truncation of the resulting spherical harmonics is calculated from the number of latitudes by:</p> $trunc = (nlat * 2 - 1) / \boxed{3}$
<b>gp2spl</b>	<p>Gridpoint to spectral (linear)</p> <p>Convert all gaussian gridpoint fields to spectral coefficients. The triangular truncation of the resulting spherical harmonics is calculated from the number of latitudes by:</p> $trunc = (nlat * 2 - 1) / \boxed{2}$ <p>Use this operator to convert ERA40 data e.g. from N80 to TL159 instead of T106.</p>
<b>sp2sp</b>	<p>Spectral to spectral</p> <p>Change the triangular truncation of all spectral fields. The operator performs downward conversion by cutting the resolution. Upward conversions are achieved by filling in zeros.</p>
<b>spcut</b>	<p>Cut spectral wave number</p> <p>Set the user defined wave numbers to zero.</p>

#### Parameter

<i>trunc</i>	INTEGER	New spectral resolution
<i>wnums</i>	INTEGER	Comma separated list of wave numbers

## Example

To transform spectral coefficients from T106 to N80 gaussian grid use:

```
cdo sp2gp ifile ofile
```

To transform spectral coefficients from TL159 to N80 gaussian grid use:

```
cdo sp2gpl ifile ofile
```

## 2.11.2. WIND - Wind transformation

### Synopsis

```
<operator> ifile ofile
```

### Description

This module converts divergence and vorticity to U and V wind and vice versa.

### Operators

- |               |  |
|---------------|--|
| <b>dv2uv</b>  | Divergence and vorticity to U and V wind<br>Calculate U and V wind on a gaussian grid from spherical harmonic coefficients of divergence and vorticity. The U and V wind must have the names u and v or the code numbers 131 and 132. The number of latitudes of the resulting gaussian grid is calculated from the triangular truncation by:<br>$nlat = NINT((trunc * \boxed{3} + 1.) / 2.)$          |
| <b>dv2uvl</b> | Divergence and vorticity to U and V wind (linear)<br>Calculate U and V wind on a gaussian grid from spherical harmonic coefficients of divergence and vorticity. The U and V wind must have the names u and v or the code numbers 131 and 132. The number of latitudes of the resulting gaussian grid is calculated from the triangular truncation by:<br>$nlat = NINT((trunc * \boxed{2} + 1.) / 2.)$ |
| <b>uv2dv</b>  | U and V wind to divergence and vorticity<br>Calculate spherical harmonic coefficients of divergence and vorticity from U and V wind. The divergence and vorticity must have the names sd and svo or code numbers 155 and 138. The triangular truncation of the resulting spherical harmonics is calculated from the number of latitudes by:<br>$trunc = (nlat * 2 - 1) / \boxed{3}$                    |
| <b>uv2dvl</b> | U and V wind to divergence and vorticity (linear)<br>Calculate spherical harmonic coefficients of divergence and vorticity from U and V wind. The divergence and vorticity must have the names sd and svo or code numbers 155 and 138. The triangular truncation of the resulting spherical harmonics is calculated from the number of latitudes by:<br>$trunc = (nlat * 2 - 1) / \boxed{2}$           |

### Example

Assume a dataset has at least spherical harmonic coefficients of divergence and vorticity. To transform the spectral divergence and vorticity to U and V wind, use:

```
cdo dv2uv ifile ofile
```

## 2.12. Formatted I/O

This section contains modules to read and write ASCII data.

Here is a short overview of all operators in this section:

<b>input</b>	ASCII input
<b>inputsrv</b>	SERVICE input
<b>inputtext</b>	EXTRA input
<b>output</b>	ASCII output
<b>outputf</b>	Formatted output
<b>outputint</b>	Integer output
<b>outputsrv</b>	SERVICE output
<b>outputtext</b>	EXTRA output

## 2.12.1. INPUT - Formatted input

### Synopsis

`input,grid ofile`

`inputsrv ofile`

`inputtext ofile`

### Description

This module reads time series of one 2D variable from standard input. All input fields must have the same horizontal grid. The format of the input depends on the actual operator.

### Operators

<b>input</b>	ASCII input Read fields with ASCII numbers from standard input and stores them in <code>ofile</code> . The numbers that are read are exactly that ones which are written out by <a href="#">output</a> .
<b>inputsrv</b>	SERVICE input Read fields with ASCII numbers from standard input and stores them in <code>ofile</code> . Each field must have a header of 8 integers (SERVICE likely). The numbers that are read are exactly that ones which are written out by <a href="#">outputsrv</a> .
<b>inputtext</b>	EXTRA input Read fields with ASCII numbers from standard input and stores them in <code>ofile</code> . Each field with a header of 4 integers (EXTRA likely). The numbers that are read are exactly that ones which are written out by <a href="#">outputtext</a> .

### Parameter

`grid`    STRING    Grid description file or name

### Example

Assume an ASCII dataset contains a field on a global regular grid with 32 longitude and 16 latitudes (512 elements). To create a GRIB dataset from the ASCII dataset use:

```
cdo -f grb input,r32x16 ofile.grb < my_ascii_data
```

## 2.12.2. OUTPUT - Formatted output

### Synopsis

```
output ifiles
outputf,format,nelem ifiles
outputint ifiles
outputsrv ifiles
outputtext ifiles
```

### Description

This module prints all values of all input datasets to standard output. All input fields must have the same horizontal grid. The format of the output depends on the actual operator.

### Operators

<b>output</b>	ASCII output Prints all values to standard output. Each row has 6 elements with the C-style format "%13.6g".
<b>outputf</b>	Formatted output Prints all values to standard output. The format and number of elements for each row must be specified by the parameters <i>format</i> and <i>nelem</i> .
<b>outputint</b>	Integer output Prints all values rounded to the nearest interger to standard output.
<b>outputsrv</b>	SERVICE output Prints all values to standard output. Each field with a header of 8 integers (SERVICE likely).
<b>outputtext</b>	EXTRA output Prints all values to standard output. Each field with a header of 4 integers (EXTRA likely).

### Parameter

<i>format</i>	STRING	C-style format for one element (e.g. %13.6g)
<i>nelem</i>	INTEGER	Number of elements for each row

### Example

To print all field elements of a dataset formatted with "%8.4g" and 8 values per line use:

```
cdo outputf,%8.4g,8 ifile
```

Example result of a dataset with one field on 64 grid points:

261.7	262	257.8	252.5	248.8	247.7	246.3	246.1
250.6	252.6	253.9	254.8	252	246.6	249.7	257.9
273.4	266.2	259.8	261.6	257.2	253.4	251	263.7
267.5	267.4	272.2	266.7	259.6	255.2	272.9	277.1
275.3	275.5	276.4	278.4	282	269.6	278.7	279.5
282.3	284.5	280.3	280.3	280	281.5	284.7	283.6
292.9	290.5	293.9	292.6	292.7	292.8	294.1	293.6
293.8	292.6	291.2	292.6	293.2	292.8	291	291.2

## 2.13. Miscellaneous

This section contains miscellaneous modules which do not fit to the other sections before.

Here is a short overview of all operators in this section:

<b>gridarea</b>	Grid cell area
<b>gridweights</b>	Grid cell weights
<b>gradsdes1</b>	GrADS data descriptor file (version 1 GRIB map)
<b>gradsdes2</b>	GrADS data descriptor file (version 2 GRIB map)
<b>smooth9</b>	9 point smoothing
<b>setrtoc</b>	Set range to constant
<b>setrtoc2</b>	Set range to constant others to constant2
<b>timsort</b>	Sort over the time
<b>const</b>	Create a constant field
<b>random</b>	Create a field with random values
<b>rotuvb</b>	Backward rotation
<b>mastrfu</b>	Mass stream function
<b>histcount</b>	Histogram count
<b>histsum</b>	Histogram sum
<b>histmean</b>	Histogram mean
<b>histfreq</b>	Histogram frequency
<b>wct</b>	Windchill temperature (C)
<b>fdns</b>	Frost days where no snow index per time period
<b>strwin</b>	Strong wind days index per time period
<b>strbre</b>	Strong breeze days index per time period
<b>strgal</b>	Strong gale days index per time period
<b>hurr</b>	Hurricane days index per time period



### 2.13.1. GRIDCELL - Grid cell quantities

#### Synopsis

```
<operator> ifile ofile
```

#### 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 description. Depending on the actual operator the grid cell area or weights are written to the output stream.

#### Operators

<b>gridarea</b>	Grid cell area Writes the grid cell area to the output stream.
<b>gridweights</b>	Grid cell weights Writes the grid cell area weights to the output stream.

#### Environment

<b>PLANET_RADIUS</b>	This variable is used to scale the computed grid cell areas to square meter. By default, PLANET_RADIUS is set to an earth radius of 6371000 meter.
----------------------	--

#### Note

## 2.13.2. GRADSDES - GrADS data descriptor file

### Synopsis

```
<operator> ifile
```

### Description

Creates a [GrADS](#) data descriptor file. Supported file formats are GRIB, SERVICE, EXTRA and IEG. For GRIB files the GrADS map file is also generated. For SERVICE and EXTRA files the grid must be specified with the CDO option '-g <grid>'. This module takes `ifile` in order to create filenames for the descriptor (`ifile.ct1`) and the map (`ifile.gmp`) file. "gradsdes" is an alias for [gradsdes2](#).

### Operators

- gradsdes1** GrADS data descriptor file (version 1 GRIB map)  
Creates a GrADS data descriptor file. Generated a machine specific version 1 GrADS map file for GRIB datasets.
- gradsdes2** GrADS data descriptor file (version 2 GRIB map)  
Creates a GrADS data descriptor file. Generated a machine independent version 2 GrADS map file for GRIB datasets. This map file can be used only with GrADS version 1.8 or newer.

### Example

To create a GrADS data descriptor file from a GRIB dataset use:

```
cdo gradsdes2 ifile.grb
```

This will create a descriptor file with the name `ifile.ct1` and the map file `ifile.gmp`. Assumed the input GRIB dataset has 3 variables over 12 time steps on a T21 grid. The contents of the resulting GrADS data description file is approximately:

```
DSET ^ ifile.grb
DTYPE GRIB
INDEX ^ ifile.gmp
XDEF 64 LINEAR 0.000000 5.625000
YDEF 32 LEVELS -85.761 -80.269 -74.745 -69.213 -63.679 -58.143
               -52.607 -47.070 -41.532 -35.995 -30.458 -24.920
               -19.382 -13.844 -8.307 -2.769 2.769 8.307
               13.844 19.382 24.920 30.458 35.995 41.532
               47.070 52.607 58.143 63.679 69.213 74.745
               80.269 85.761
ZDEF 4 LEVELS 925 850 500 200
TDEF 12 LINEAR 12:00 Z1jan1987 1mo
TITLE ifile.grb T21 grid
OPTIONS yrev
UNDEF -9e+33
VARS 3
geosp 0 129,1,0 surface geopotential (orography) [m^2/s^2]
t      4 130,99,0 temperature [K]
tslm1 0 139,1,0 surface temperature of land [K]
ENDVARS
```

### 2.13.3. SMOOTH9 - 9 point smoothing

#### Synopsis

```
smooth9 ifile ofile
```

#### Description

Performs a 9 point smoothing on all fields with a rectangular 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.

### 2.13.4. SETRANGE - Set range to constant

#### Synopsis

```
setrtoc,rmin,rmax,c ifile ofile
```

```
setrtoc2,rmin,rmax,c,c2 ifile ofile
```

#### Description

This module sets part of a field to a constant value or missing values. Which part of the field is set depends on the actual operator.

#### Operators

**setrtoc**      Set range to constant

$$o(t, x) = \begin{cases} c & \text{if } i(t, x) \geq rmin \wedge i(t, x) \leq rmax \\ i(t, x) & \text{if } i(t, x) < rmin \vee i(t, x) > rmax \end{cases}$$

**setrtoc2**    Set range to constant others to constant2

$$o(t, x) = \begin{cases} c & \text{if } i(t, x) \geq rmin \wedge i(t, x) \leq rmax \\ c2 & \text{if } i(t, x) < rmin \vee i(t, x) > rmax \end{cases}$$

#### Parameter

<i>rmin</i>	FLOAT	Lower bound
<i>rmax</i>	FLOAT	Upper bound
<i>c</i>	FLOAT	New value - inside of range
<i>c2</i>	FLOAT	New value - outside of range

### 2.13.5. TIMSORT - Timsort

#### Synopsis

```
timsort ifile ofile
```

#### Description

Sorts for every field position the elements in ascending order over all time steps. After sorting it is:

$$o(t_1, x) < o(t_2, x) \quad \forall (t_1 < t_2), x$$

#### Example

To sort all field elements of a dataset over all time steps use:

```
cdo timsort ifile ofile
```

### 2.13.6. VARGEN - Generate a field

#### Synopsis

```
const,const,grid ofile
random,grid ofile
```

#### Description

Generates a dataset with one field. The size of the field is specified by the user given grid description. According to the actual operator all field elements are constant or filled with random numbers.

#### Operators

<b>const</b>	Create a constant field Creates a constant field. All field elements of the grid have the same value.
<b>random</b>	Create a field with random values Creates a field with rectangularly distributed random numbers in the interval [0,1].

#### Parameter

<i>const</i>	FLOAT	Constant
<i>grid</i>	STRING	Target grid description file or name

### 2.13.7. ROTUV - Rotation

#### Synopsis

```
rotuvb,u,v,... ifile ofile
```

#### Description

This is a special operator for datasets 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.

#### Parameter

<code>u,v,...</code>	STRING	Pairs of zonal and meridional velocity components (use variable names or code numbers)
----------------------	--------	--

#### Example

To transform the u and v velocity of a dataset from a rotated spherical system to a geographical system use:

```
cdo rotuvb,u,v ifile ofile
```

### 2.13.8. MASTRFU - Mass stream function

#### Synopsis

```
mastrfu ifile ofile
```

#### Description

This is a special operator for the post processing of the atmospheric general circulation model [ECHAM](#). It computes the mass stream function (code number 272). The input dataset must be a zonal mean of v-velocity (code number 132) on pressure levels.

#### Example

To compute the mass stream function from a zonal mean v-velocity dataset use:

```
cdo mastrfu ifile ofile
```

### 2.13.9. HISTOGRAM - Histogram

#### Synopsis

```
<operator> , bounds ifile ofile
```

#### Description

This module creates bins for a histogram of the input data. The bins must 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.

#### Operators

<b>histcount</b>	Histogram count Number of elements in the bin range.
<b>histsum</b>	Histogram sum Sum of elements in the bin range.
<b>histmean</b>	Histogram mean Mean of elements in the bin range.
<b>histfreq</b>	Histogram frequency Frequency of elements in the bin range.

#### Parameter

<i>bounds</i>	FLOAT	Comma separated list of the bin bounds (-inf and inf valid)
---------------	-------	---

### 2.13.10. WCT - Windchill temperature (C)

#### Synopsis

```
wct ifile1 ifile2 ofile
```

#### Description

Let `ifile1` and `ifile2` be time series of temperature and wind speed records, then a corresponding time series of resulting windchill temperatures is written to `ofile`. The wind chill temperature calculation is only valid for a temperature of  $T \leq 33$  C and a wind speed of  $v \geq 1.39$  m/s. Whenever these conditions are not satisfied, a missing value is written to `@ofile`. Note that temperature and wind speed records must be given in units of C and m/s, respectively.

### 2.13.11. FDNS - Frost days where no snow index per time period

#### Synopsis

```
fdns ifile1 ifile2 ofile
```

#### Description

Let `ifile1` be a time series of daily minimum temperatures `TN` and `ifile2` be a corresponding series of daily surface snow amounts. Then counted is the number of days where  $TN < 0$  Celsius and the surface snow amount is less than 1 cm. The temperature `TN` must be given in units of Kelvin. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

### 2.13.12. STRWIN - Strong wind days index per time period

#### Synopsis

```
strwin[,v] ifile ofile
```

#### Description

Let `ifile` be a time series of daily maximum horizontal wind speeds `VX`, then counted is the number of days where  $VX > v$ . 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  $v$  must 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 for a time step in `ofile` is the date of the last contributing time step in `ifile`.

#### Parameter

<code>v</code>	FLOAT	Horizontal wind speed threshold (m/s, default $v = 10.5$ m/s)
----------------	-------	---

### 2.13.13. STRBRE - Strong breeze days index per time period

#### Synopsis

```
strbre ifile ofile
```

#### Description

Let `ifile` be a time series of daily maximum horizontal wind speeds  $VX$ , then counted is the number of days where  $VX$  is greater than or equal to 10.5 m/s. 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 must be given in units of m/s. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

### 2.13.14. STRGAL - Strong gale days index per time period

#### Synopsis

```
strgal ifile ofile
```

#### Description

Let `ifile` be a time series of daily maximum horizontal wind speeds  $VX$ , then counted is the number of days where  $VX$  is greater than or equal to 20.5 m/s. 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 must be given in units of m/s. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

### 2.13.15. HURR - Hurricane days index per time period

#### Synopsis

```
hurr ifile ofile
```

#### Description

Let `ifile` be a time series of daily maximum horizontal wind speeds  $VX$ , then counted is the number of days where  $VX$  is greater than or equal to 32.5 m/s. 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 must be given in units of m/s. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.



## 2.14. Climate indices

This section contains modules to compute the climate indices of daily temperature and precipitation extremes.

Here is a short overview of all operators in this section:

<a href="#">eca_cdd</a>	Consecutive dry days index per time period
<a href="#">eca_cfd</a>	Consecutive frost days index per time period
<a href="#">eca_csu</a>	Consecutive summer days index per time period
<a href="#">eca_cwd</a>	Consecutive wet days index per time period
<a href="#">eca_cwdi</a>	Cold wave duration index wrt mean of reference period
<a href="#">eca_cwfi</a>	Cold-spell days index wrt 10th percentile of reference period
<a href="#">eca_etr</a>	Intra-period extreme temperature range
<a href="#">eca_fd</a>	Frost days index per time period
<a href="#">eca_gsl</a>	Growing season length index
<a href="#">eca_hd</a>	Heating degree days per time period
<a href="#">eca_hwdi</a>	Heat wave duration index wrt mean of reference period
<a href="#">eca_hwfi</a>	Warm spell days index wrt 90th percentile of reference period
<a href="#">eca_id</a>	Ice days index per time period
<a href="#">eca_r10mm</a>	Heavy precipitation days index per time period
<a href="#">eca_r20mm</a>	Very heavy precipitation days index per time period
<a href="#">eca_r75p</a>	Moderate wet days wrt 75th percentile of reference period
<a href="#">eca_r75ptot</a>	Precipitation percent due to R75p days
<a href="#">eca_r90p</a>	Wet days wrt 90th percentile of reference period
<a href="#">eca_r90ptot</a>	Precipitation percent due to R90p days
<a href="#">eca_r95p</a>	Very wet days wrt 95th percentile of reference period
<a href="#">eca_r95ptot</a>	Precipitation percent due to R95p days
<a href="#">eca_r99p</a>	Extremely wet days wrt 99th percentile of reference period
<a href="#">eca_r99ptot</a>	Precipitation percent due to R99p days
<a href="#">eca_rr1</a>	Wet days index per time period
<a href="#">eca_rx1day</a>	Highest one day precipitation amount per time period
<a href="#">eca_rx5day</a>	Highest five-day precipitation amount per time period
<a href="#">eca_sdii</a>	Simple daily intensity index per time period

---

<b>eca_su</b>	Summer days index per time period
<b>eca_tg10p</b>	Cold days percent wrt 10th percentile of reference period
<b>eca_tg90p</b>	Warm days percent wrt 90th percentile of reference period
<b>eca_tn10p</b>	Cold nights percent wrt 10th percentile of reference period
<b>eca_tn90p</b>	Warm nights percent wrt 90th percentile of reference period
<b>eca_tr</b>	Tropical nights index per time period
<b>eca_tx10p</b>	Very cold days percent wrt 10th percentile of reference period
<b>eca_tx90p</b>	Very warm days percent wrt 90th percentile of reference period

### 2.14.1. ECACDD - Consecutive dry days index per time period

#### Synopsis

```
eca_cdd ifile ofile
```

#### Description

Let `ifile` be a time series of daily precipitation amounts `RR`, then counted is the largest number of consecutive days where `RR` is less than 1 mm. A further output variable is the number of dry periods of more than 5 days. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

#### Example

To get the largest number of consecutive dry days for a time series of daily precipitation amounts, use:

```
cdo eca_cdd rrfile ofile
```

### 2.14.2. ECACFD - Consecutive frost days index per time period

#### Synopsis

```
eca_cfd ifile ofile
```

#### Description

Let `ifile` be a time series of daily minimum temperatures `TN`, then counted is the largest number of consecutive days where  $TN < 0$  Celsius. Note that `TN` must be given in units of Kelvin. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

#### Example

To get the largest number of consecutive frost days for a time series of daily minimum temperatures, use:

```
cdo eca_cfd tnfile ofile
```

### 2.14.3. ECACSU - Consecutive summer days index per time period

#### Synopsis

```
eca_csu[,T] ifile ofile
```

#### Description

Let `ifile` be a time series of daily maximum temperatures TX, then counted is the largest number of consecutive days where TX > T. The number T is an optional parameter with default T = 25 Celsius. Note that TN must be given in units of Kelvin, whereas T must be given in degrees Celsius. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

#### Parameter

<code>T</code>	FLOAT	Temperature threshold ( Celsius, default: T = 25 Celsius)
----------------	-------	---

#### Example

To get the largest number of consecutive summer days for a time series of daily minimum temperatures, use:

```
cdo eca_csu txfile ofile
```

### 2.14.4. ECACWD - Consecutive wet days index per time period

#### Synopsis

```
eca_cwd ifile ofile
```

#### Description

Let `ifile` be a time series of daily precipitation amounts RR, then counted is the largest number of consecutive days where RR is at least 1 mm. A further output variable is the number of wet periods of more than 5 days. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

#### Example

To get the largest number of consecutive wet days for a time series of daily precipitation amounts, use:

```
cdo eca_cwd rrfile ofile
```

### 2.14.5. ECACWDI - Cold wave duration index wrt mean of reference period

#### Synopsis

```
eca_cwdi[,nday[,T]] ifile1 ifile2 ofile
```

#### Description

Let `ifile1` be a time series of daily minimum temperatures `TN`, and let `ifile2` 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` Celsius. A further output variable is the number of cold waves longer than or equal to `nday` days. Note that both `TN` and `TNnorm` must be given in the same units. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

#### Parameter

<code>nday</code>	INTEGER	Number of consecutive days (default: <code>nday = 6</code> )
<code>T</code>	FLOAT	Temperature offset ( Celsius, default: <code>T = 5</code> Celsius)

#### Example

To compute the cold wave duration index for a time series of daily minimum temperatures, use:

```
cdo eca_cwdi tfile tnnormfile ofile
```

### 2.14.6. ECACWFI - Cold-spell days index wrt 10th percentile of reference period

#### Synopsis

```
eca_cwfi[,nday] ifile1 ifile2 ofile
```

#### Description

Let `ifile1` be a time series of daily mean temperatures `TG`, and `ifile2` 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. Note that both `TG` and `TGn10` must be given in the same units. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

#### Parameter

<code>nday</code>	INTEGER	Number of consecutive days (default: <code>nday = 6</code> )
-------------------	---------	--

#### Example

To compute the number of cold-spell days for a time series of daily mean temperatures, use:

```
cdo eca_cwfi tgfile tgn10file ofile
```

### 2.14.7. ECAETR - Intra-period extreme temperature range

#### Synopsis

```
eca_etr ifile1 ifile2 ofile
```

#### Description

Let `ifile1` and `ifile2` be time series of maximum and minimum temperatures 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 must be given in the same units. The date information for a time step in `ofile` is the date of the last contributing time steps in `ifile1` and `ifile2`.

#### Example

To get the intra-period extreme temperature range for two time series of maximum and minimum temperatures, use:

```
cdo eca_etr txfile tnfile ofile
```

### 2.14.8. ECAFD - Frost days index per time period

#### Synopsis

```
eca_fd ifile ofile
```

#### Description

Let `ifile` be a time series of daily minimum temperatures TN, then counted is the number of days where  $TN < 0$  Celsius. Note that TN must be given in units of Kelvin. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

#### Example

To get the number of frost days for a time series of daily minimum temperatures, use:

```
cdo eca_fd tnfile ofile
```

### 2.14.9. ECAGSL - Growing season length index

#### Synopsis

```
eca_gsl[,nday[,T[,fland]]] ifile1 ifile2 ofile
```

#### Description

Let `ifile1` be a time series of daily mean temperatures `TG`, and `ifile2` be a land-water mask. Then counted are the number of days between the first occurrence of at least `nday` consecutive days with  $TG > T$  and the first occurrence after 1 July of at least `nday` consecutive days with  $TG < T$ . The numbers `nday` and `T` are optional parameter with default `nday = 6` and `T = 5` Celsius. The number `fland` is an optional parameter with default value `fland = 0.5` and denotes the fraction of a grid point that must 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` must be given in units of Kelvin, whereas `T` must be given in degrees Celsius. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

#### Parameter

<code>nday</code>	INTEGER	Number of consecutive days (default: <code>nday = 6</code> )
<code>T</code>	FLOAT	Temperature threshold ( Celsius, default: <code>T = 5</code> Celsius)
<code>fland</code>	FLOAT	Land fraction threshold (default: <code>fland = 0.5</code> )

#### Example

To get the growing season length for a time series of daily mean temperatures, use:

```
cdo eca_gsl tgfile maskfile ofile
```

### 2.14.10. ECAHD - Heating degree days per time period

#### Synopsis

```
eca_hd[,T1[,T2]] ifile ofile
```

#### Description

Let `ifile` be a time series of daily mean temperatures `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 Celsius is used for both parameters. If only `T1` is given, `T2` is set to `T1`. Note that `TG` must be given in units of kelvin, whereas `T1` and `T2` must be given in degrees Celsius. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

#### Parameter

<code>T1</code>	FLOAT	Temperature limit ( Celsius, default: <code>T1 = 17 Celsius</code> )
<code>T2</code>	FLOAT	Temperature limit ( Celsius, default: <code>T2 = T1</code> )

#### Example

To compute the heating degree days for a time series of daily mean temperatures, use:

```
cdo eca_hd tgfile ofile
```

### 2.14.11. ECAHWDI - Heat wave duration index wrt mean of reference period

#### Synopsis

```
eca_hwdi[,nday[,T]] ifile1 ifile2 ofile
```

#### Description

Let `ifile1` be a time series of daily maximum temperatures `TX`, and let `ifile2` 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 Celsius`. A further output variable is the number of heat waves longer than or equal to `nday` days. Note that both `TX` and `TXnorm` must be given in the same units. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

#### Parameter

<code>nday</code>	INTEGER	Number of consecutive days (default: <code>nday = 6</code> )
<code>T</code>	FLOAT	Temperature offset ( Celsius, default: <code>T = 5 Celsius</code> )

#### Example

To compute the heat wave duration index for a time series of daily maximum temperatures, use:

```
cdo eca_hwdi txfile txnormfile ofile
```



## 2.14.12. ECAHWFI - Warm spell days index wrt 90th percentile of reference period

### Synopsis

```
eca_hwfi[,nday] ifile1 ifile2 ofile
```

### Description

Let `ifile1` be a time series of daily mean temperatures TG, and `ifile2` 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. Note that both TG and TGn90 must be given in the same units. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

### Parameter

`nday`     INTEGER     Number of consecutive days (default: `nday = 6`)

### Example

To compute the number of warm-spell days for a time series of daily mean temperatures, use:

```
cdo eca_hwfi tgfile tgn90file ofile
```

## 2.14.13. ECAID - Ice days index per time period

### Synopsis

```
eca_id ifile ofile
```

### Description

Let `ifile` be a time series of daily maximum temperatures TX, then counted is the number of days where  $TX < 0$  Celsius. Note that TX must be given in units of Kelvin. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

### Example

To get the number of ice days for a time series of daily maximum temperatures, use:

```
cdo eca_id txfile ofile
```

#### 2.14.14. ECAR10MM - Heavy precipitation days index per time period

##### Synopsis

```
eca_r10mm ifile ofile
```

##### Description

Let `ifile` be a time series of daily precipitation amounts `RR`, then counted is the number of days where `RR` is at least 10 mm. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

##### Example

To get the number of days with precipitation greater than 10 mm for a time series of daily precipitation amounts, use:

```
cdo eca_r10mm rfile ofile
```

#### 2.14.15. ECAR20MM - Very heavy precipitation days index per time period

##### Synopsis

```
eca_r20mm ifile ofile
```

##### Description

Let `ifile` be a time series of daily precipitation amounts `RR`, then counted is the number of days where `RR` is at least 20 mm. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

##### Example

To get the number of days with precipitation greater than 20 mm for a time series of daily precipitation amounts, use:

```
cdo eca_r20mm rfile ofile
```

## 2.14.16. ECAR75P - Moderate wet days wrt 75th percentile of reference period

### Synopsis

```
eca_r75p ifile1 ifile2 ofile
```

### Description

Let `ifile1` be a time series of daily precipitation amounts `RR`, and `ifile2` be the 75th percentile `RRn75` of daily precipitation amounts at wet days for any period used as reference. Then calculated is the percentage of wet days with  $RR > RRn75$ . The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

### Example

To compute the percentage of wet days where the daily precipitation amount is greater than the 75th percentile of the daily precipitation amount at wet days for a given reference period, use:

```
cdo eca_r75p rrfile rrn75file ofile
```

## 2.14.17. ECAR75PTOT - Precipitation percent due to R75p days

### Synopsis

```
eca_r75ptot ifile1 ifile2 ofile
```

### Description

Let `ifile1` be a time series of daily precipitation amounts `RR`, and `ifile2` be the 75th percentile `RRn75` of daily precipitation amounts at wet days for any period used as reference. Then calculated is the ratio of the precipitation sum at wet days with  $RR > RRn75$  to the total precipitation sum. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

## 2.14.18. ECAR90P - Wet days wrt 90th percentile of reference period

### Synopsis

```
eca_r90p ifile1 ifile2 ofile
```

### Description

Let `ifile1` be a time series of daily precipitation amounts `RR`, and `ifile2` be the 90th percentile `RRn90` of daily precipitation amounts at wet days for any period used as reference. Then calculated is the percentage of wet days with  $RR > RRn90$ . The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

### Example

To compute the percentage of wet days where the daily precipitation amount is greater than the 90th percentile of the daily precipitation amount at wet days for a given reference period, use:

```
cdo eca_r90p rfile rrn90file ofile
```

## 2.14.19. ECAR90PTOT - Precipitation percent due to R90p days

### Synopsis

```
eca_r90ptot ifile1 ifile2 ofile
```

### Description

Let `ifile1` be a time series of daily precipitation amounts `RR`, and `ifile2` be the 90th percentile `RRn90` of daily precipitation amounts at wet days for any period used as reference. Then calculated is the ratio of the precipitation sum at wet days with  $RR > RRn90$  to the total precipitation sum. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

## 2.14.20. ECAR95P - Very wet days wrt 95th percentile of reference period

### Synopsis

```
eca_r95p ifile1 ifile2 ofile
```

### Description

Let `ifile1` be a time series of daily precipitation amounts `RR`, and `ifile2` be the 95th percentile `RRn95` of daily precipitation amounts at wet days for any period used as reference. Then calculated is the percentage of wet days with  $RR > RRn95$ . The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

### Example

To compute the percentage of wet days where the daily precipitation amount is greater than the 95th percentile of the daily precipitation amount at wet days for a given reference period, use:

```
cdo eca_r95p rrfile rrn95file ofile
```

## 2.14.21. ECAR95PTOT - Precipitation percent due to R95p days

### Synopsis

```
eca_r95ptot ifile1 ifile2 ofile
```

### Description

Let `ifile1` be a time series of daily precipitation amounts `RR`, and `ifile2` be the 95th percentile `RRn95` of daily precipitation amounts at wet days for any period used as reference. Then calculated is the ratio of the precipitation sum at wet days with  $RR > RRn95$  to the total precipitation sum. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

## 2.14.22. ECAR99P - Extremely wet days wrt 99th percentile of reference period

### Synopsis

```
eca_r99p ifile1 ifile2 ofile
```

### Description

Let `ifile1` be a time series of daily precipitation amounts `RR`, and `ifile2` be the 99th percentile `RRn99` of daily precipitation amounts at wet days for any period used as reference. Then calculated is the percentage of wet days with  $RR > RRn99$ . The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

### Example

To compute the percentage of wet days where the daily precipitation amount is greater than the 99th percentile of the daily precipitation amount at wet days for a given reference period, use:

```
cdo eca_r99p rfile rrn99file ofile
```

## 2.14.23. ECAR99PTOT - Precipitation percent due to R99p days

### Synopsis

```
eca_r99ptot ifile1 ifile2 ofile
```

### Description

Let `ifile1` be a time series of daily precipitation amounts `RR`, and `ifile2` be the 99th percentile `RRn99` of daily precipitation amounts at wet days for any period used as reference. Then calculated is the ratio of the precipitation sum at wet days with  $RR > RRn99$  to the total precipitation sum. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

## 2.14.24. ECARR1 - Wet days index per time period

### Synopsis

```
eca_rr1 ifile ofile
```

### Description

Let `ifile` be a time series of daily precipitation amounts `RR`, then counted is the number of days where `RR` is at least 1 mm. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

### Example

To get the number of wet days for a time series of daily precipitation amounts, use:

```
cdo eca_rr1 rrfile ofile
```

## 2.14.25. ECARX1DAY - Highest one day precipitation amount per time period

### Synopsis

```
eca_rx1day[,mode] ifile ofile
```

### Description

Let `ifile` be a time series of daily precipitation amounts `RR`, then the maximum of `RR` is written to `ofile`. If the optional parameter `mode` is set to 'm', then maximum daily precipitation amounts are determined for each month. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

### Parameter

<i>mode</i>	STRING	Operation mode (optional). If <code>mode = 'm'</code> then maximum daily precipitation amounts are determined for each month
-------------	--------	--

### Example

To get the maximum of a time series of daily precipitation amounts, use:

```
cdo eca_rx1day rrfile ofile
```

If you are interested in the maximum daily precipitation for each month, use:

```
cdo eca_rx1day,m rrfile ofile
```

Apart from metadata information, both operations yield the same as:

```
cdo timmax rrfile ofile  
cdo monmax rrfile ofile
```

## 2.14.26. ECARX5DAY - Highest five-day precipitation amount per time period

### Synopsis

```
eca_rx5day[,x] ifile ofile
```

### Description

Let `ifile` be a time series of 5-day precipitation totals `RR`, then the maximum of `RR` is written to `ofile`. 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. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

### Parameter

<code>x</code>	FLOAT	Precipitation threshold (mm, default: <code>x = 50</code> mm)
----------------	-------	---

### Example

To get the maximum of a time series of 5-day precipitation totals, use:

```
cdo eca_rx5day rrfile ofile
```

Apart from metadata information, the above operation yields the same as:

```
cdo timmax rrfile ofile
```

## 2.14.27. ECASDII - Simple daily intensity index per time period

### Synopsis

```
eca_sdii ifile ofile
```

### Description

Let `ifile` be a time series of daily precipitation amounts `RR`, then the mean precipitation amount at wet days ( $RR > 1$  mm) is written to `ofile`. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

### Example

To get the daily intensity index for a time series of daily precipitation amounts, use:

```
cdo eca_sdii rrfile ofile
```

## 2.14.28. ECASU - Summer days index per time period

### Synopsis

```
eca_su[,T] ifile ofile
```



## Description

Let `ifile` be a time series of daily maximum temperatures `TX`, then counted is the number of days where `TX > T`. The number `T` is an optional parameter with default `T = 25 Celsius`. Note that `TX` must be given in units of Kelvin, whereas `T` must be given in degrees Celsius. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

## Parameter

<code>T</code>	<code>FLOAT</code>	Temperature threshold ( Celsius, default: <code>T = 25 Celsius</code> )
----------------	--------------------	---

## Example

To get the number of summer days for a time series of daily maximum temperatures, use:

```
cdo eca_su txfile ofile
```

### 2.14.29. ECATG10P - Cold days percent wrt 10th percentile of reference period

#### Synopsis

```
eca_tg10p ifile1 ifile2 ofile
```

#### Description

Let `ifile1` be a time series of daily mean temperatures TG, and `ifile2` be the 10th percentile TGn10 of daily mean temperatures for any period used as reference. Then calculated is the percentage of time where  $TG < TGn10$ . Note that both TG and TGn10 must be given in the same units. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

#### Example

To compute the percentage of time where the daily mean temperature is less than the 10th percentile of the daily mean temperature for a given reference period, use:

```
cdo eca_tg10p tgfile tgn10file ofile
```

### 2.14.30. ECATG90P - Warm days percent wrt 90th percentile of reference period

#### Synopsis

```
eca_tg90p ifile1 ifile2 ofile
```

#### Description

Let `ifile1` be a time series of daily mean temperatures TG, and `ifile2` be the 90th percentile TGn90 of daily mean temperatures for any period used as reference. Then calculated is the percentage of time where  $TG > TGn90$ . Note that both TG and TGn90 must be given in the same units. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

#### Example

To compute the percentage of time where the daily mean temperature is greater than the 90th percentile of the daily mean temperature for a given reference period, use:

```
cdo eca_tg90p tgfile tgn90file ofile
```

### 2.14.31. ECATN10P - Cold nights percent wrt 10th percentile of reference period

#### Synopsis

```
eca_tn10p ifile1 ifile2 ofile
```

#### Description

Let `ifile1` be a time series of daily minimum temperatures `TN`, and `ifile2` be the 10th percentile `TNn10` of daily minimum temperatures for any period used as reference. Then calculated is the percentage of time where  $TN < TNn10$ . Note that both `TN` and `TNn10` must be given in the same units. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

#### Example

To compute the percentage of time where the daily minimum temperature is less than the 10th percentile of the daily minimum temperature for a given reference period, use:

```
cdo eca_tn10p tnfile tnn10file ofile
```

### 2.14.32. ECATN90P - Warm nights percent wrt 90th percentile of reference period

#### Synopsis

```
eca_tn90p ifile1 ifile2 ofile
```

#### Description

Let `ifile1` be a time series of daily minimum temperatures `TN`, and `ifile2` be the 90th percentile `TNn90` of daily minimum temperatures for any period used as reference. Then calculated is the percentage of time where  $TN > TNn90$ . Note that both `TN` and `TNn90` must be given in the same units. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

#### Example

To compute the percentage of time where the daily minimum temperature is greater than the 90th percentile of the daily minimum temperature for a given reference period, use:

```
cdo eca_tn90p tnfile tnn90file ofile
```

### 2.14.33. ECATR - Tropical nights index per time period

#### Synopsis

```
eca_tr[,T] ifile ofile
```

#### Description

Let `ifile` be a time series of daily minimum temperatures `TN`, then counted is the number of days where  $TN > T$ . The number `T` is an optional parameter with default  $T = 20$  Celsius. Note that `TN` must be given in units of Kelvin, whereas `T` must be given in degrees Celsius. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile`.

#### Parameter

<code>T</code>	FLOAT	Temperature threshold ( Celsius, default: $T = 20$ Celsius)
----------------	-------	---

#### Example

To get the number of tropical nights for a time series of daily minimum temperatures, use:

```
cdo eca_tr tnfile ofile
```

### 2.14.34. ECATX10P - Very cold days percent wrt 10th percentile of reference period

#### Synopsis

```
eca_tx10p ifile1 ifile2 ofile
```

#### Description

Let `ifile1` be a time series of daily maximum temperatures `TX`, and `ifile2` be the 10th percentile `TNx10` of daily maximum temperatures for any period used as reference. Then calculated is the percentage of time where  $TX < TX_{n10}$ . Note that both `TX` and `TNx10` must be given in the same units. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

#### Example

To compute the percentage of time where the daily maximum temperature is less than the 10th percentile of the daily maximum temperature for a given reference period, use:

```
cdo eca_tx10p txfile txn10file ofile
```

### 2.14.35. ECATX90P - Very warm days percent wrt 90th percentile of reference period

#### Synopsis

```
eca_tx90p ifile1 ifile2 ofile
```

#### Description

Let `ifile1` be a time series of daily maximum temperatures TX, and `ifile2` be the 90th percentile TXn90 of daily maximum temperatures for any period used as reference. Then calculated is the percentage of time where  $TX > TXn90$ . Note that both TX and TXn90 must be given in the same units. The date information for a time step in `ofile` is the date of the last contributing time step in `ifile1`.

#### Example

To compute the percentage of time where the daily maximum temperature is greater than the 90th percentile of the daily maximum temperature for a given reference period, use:

```
cdo eca_tx90p txfile txn90file ofile
```

# Bibliography

[CDI]

[Climate Data Interface](#), from the [Max Planck Institute for Meteorologie](#)

[ECHAM]

The atmospheric general circulation model ECHAM5, from the [Max Planck Institute for Meteorologie](#)

[GrADS]

[Grid Analysis and Display System](#), from the Center for Ocean-Land-Atmosphere Studies ([COLA](#))

[GRIB]

[GRIB version 1](#), from the World Meteorological Organisation ([WMO](#))

[INTERA]

[INTERA Software Package](#), from the [Max Planck Institute for Meteorologie](#)

[netCDF]

[NetCDF Software Package](#), from the [UNIDATA](#) Program Center of the University Corporation for Atmospheric Research

[PINGO]

The [PINGO](#) package, from the [Model & Data group](#) at the Max Planck Institute for Meteorologie

[SCRIP]

[SCRIP Software Package](#), from the Los Alamos National Laboratory

## A. Hints for PINGO user

Some **CDO** operators have the same name as in PINGO but the meaning is different. The following table gives an overview of those operators.

Operator name	<b>CDO</b>	PINGO
min	Minimum of two fields	Time minimum
max	Maximum of two fields	Time maximum
daymean	Daily mean	Multi-year daily mean
daymin	Daily minimum	Multi-year daily minimum
daymax	Daily maximum	Multi-year daily maximum
monmean	Monthly mean	Multi-year monthly mean
monmin	Monthly minimum	Multi-year monthly minimum
monmax	Monthly maximum	Multi-year monthly maximum
seasmean	Seasonally mean	Multi-year seasonally mean

There are also some **CDO** operators with the same functionality as in PINGO but the name is different. The following table gives an overview of those operators.

	<b>CDO</b>	PINGO
Maximum of two fields	max	max2
Minimum of two fields	min	min2
Field mean, min, max	fldmean, fldmin, fldmax	meanr minr, maxr
Time mean, min, max	timmean, timmin, timmax	mean, min, max
Daily mean, min, max	daymean, daymin, daymax	daymeans, daymins, daymaxs
Monthly mean, min, max	monmean, monmin, monmax	monmeans, monmins, monmaxs
Yearly mean, min, max	yearmean, yearmin, yearmax	yearmeans, yearmins, yearmaxs
Running mean	runmean	runmeans
Seasonally mean	seasmean	seasmeans
Multi-year daily mean	ydaymean	daymean
Multi-year monthly mean	ymonmean	monmean
Multi-year seasonally mean	yseasmean	seasmean

## B. Grid description examples

### B.1. Example of a curvilinear grid description

Here is an example for the **CDO** description of a curvilinear grid. `xvals/yvals` describes the position of the 6x5 quadrilateral grid cells. The first 4 values of `xbounds/ybounds` are the corners of the first grid cell.

```

gridtype : curvilinear
gridsize : 30
xsize    : 6
ysize    : 5
xvals     : -21  -11   0   11   21   30  -25  -13   0   13
             25   36  -31  -16   0   16   31   43  -38  -21
             0    21   38   52  -51  -30   0    30   51   64
xbounds    : -23  -14  -17  -28          -14  -5   -6  -17          -5   5   6  -6
             5    14   17   6           14   23   28   17          23   32   38   28
             -28  -17  -21  -34          -17  -6   -7  -21          -6   6   7   -7
             6    17   21   7           17   28   34   21          28   38   44   34
             -34  -21  -27  -41          -21  -7   -9  -27          -7   7   9   -9
             7    21   27   9           21   34   41   27          34   44   52   41
             -41  -27  -35  -51          -27  -9  -13  -35          -9   9   13  -13
             9    27   35   13          27   41   51   35          41   52   63   51
             -51  -35  -51  -67          -35  -13  -21  -51          -13  13   21  -21
             13   35   51   21          35   51   67   51          51   63   77   67
yvals      : 29   32   32   32   29   26   39   42   42   42
             39   35   48   51   52   51   48   43   57   61
             62   61   57   51   65   70   72   70   65   58
ybounds    : 23   26   36   32          26   27   37   36          27   27   37   37
             27   26   36   37          26   23   32   36          23   19   28   32
             32   36   45   41          36   37   47   45          37   37   47   47
             37   36   45   47          36   32   41   45          32   28   36   41
             41   45   55   50          45   47   57   55          47   47   57   57
             47   45   55   57          45   41   50   55          41   36   44   50
             50   55   64   58          55   57   67   64          57   57   67   67
             57   55   64   67          55   50   58   64          50   44   51   58
             58   64   72   64          64   67   77   72          67   67   77   77
             67   64   72   77          64   58   64   72          58   51   56   64

```

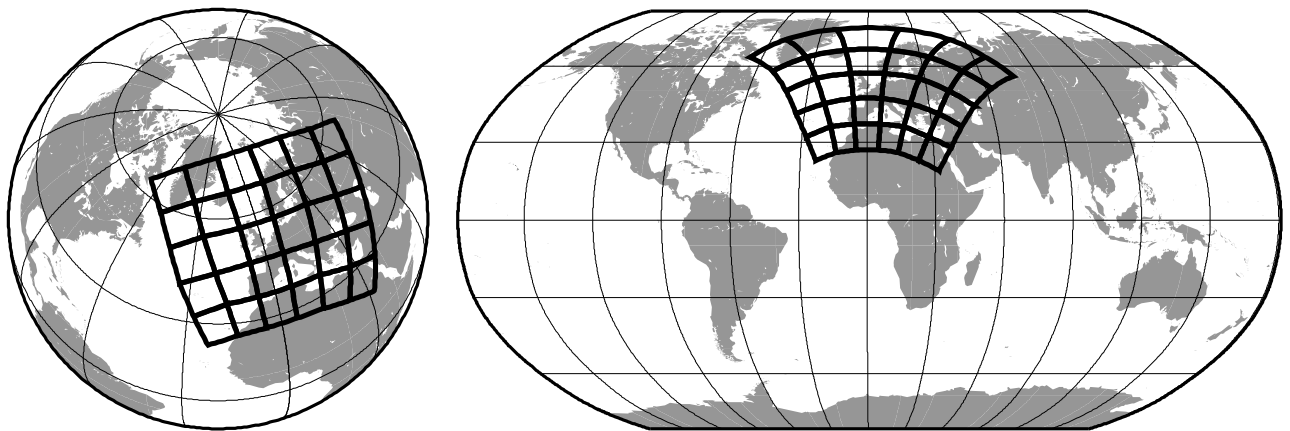


Figure B.1.: Orthographic and Robinson projection of the curvilinear grid



## B.2. Example description for unstructured grid cells

Here is an example of the **CDO** description for unstructured grid cells. xvals/yvals describes the position of 30 independent hexagonal grid cells. The first 6 values of xbounds/ybounds are the corners of the first grid cell.

```

gridtype : cell
gridsize : 30
nvertex  : 6
xvals    : -36  36   0  -18  18  108  72  54  90  180
           144 126 162 -108 -144 -162 -126 -72 -90 -54
           0   72  36  144 108 -144 180 -72 -108 -36
xbounds   : 339   0   0  288 288 309   21  51  72  72   0   0
           0   16  21   0 339 344   340   0  -0  344 324 324
           20  36  36  16   0   0   93 123 144 144  72  72
           72  88  93  72  51  56   52  72  72  56  36  36
           92 108 108  88  72  72   165 195 216 216 144 144
           144 160 165 144 123 128   124 144 144 128 108 108
           164 180 180 160 144 144   237 267 288 288 216 216
           216 232 237 216 195 200   196 216 216 200 180 180
           236 252 252 232 216 216   288 304 309 288 267 272
           268 288 288 272 252 252   308 324 324 304 288 288
           345 324 324  36  36  15    36  36 108 108  87  57
           20  15  36  57  52  36   108 108 180 180 159 129
           92  87 108 129 124 108   180 180 252 252 231 201
           164 159 180 201 196 180   252 252 324 324 303 273
           236 231 252 273 268 252   308 303 324 345 340 324
yvals     :  58  58  32   0   0  58  32   0   0  58
           32   0   0  58  32   0   0  32   0   0
           -58 -58 -32 -58 -32 -58 -32 -58 -32 -32
ybounds   :  41  53  71  71  53  41   41  41  53  71  71  53
           11  19  41  53  41  19   -19 -7  11  19  7 -11
           -19 -11  7  19  11 -7    41  41  53  71  71  53
           11  19  41  53  41  19   -19 -7  11  19  7 -11
           -19 -11  7  19  11 -7    41  41  53  71  71  53
           11  19  41  53  41  19   -19 -7  11  19  7 -11
           -19 -11  7  19  11 -7    11  19  41  53  41  19
           -19 -7  11  19  7 -11   -19 -11  7  19  11 -7
           -41 -53 -71 -71 -53 -41   -53 -71 -71 -53 -41 -41
           -19 -41 -53 -41 -19 -11   -53 -71 -71 -53 -41 -41
           -19 -41 -53 -41 -19 -11   -53 -71 -71 -53 -41 -41
           -19 -41 -53 -41 -19 -11   -53 -71 -71 -53 -41 -41
           -19 -41 -53 -41 -19 -11   -19 -41 -53 -41 -19 -11

```

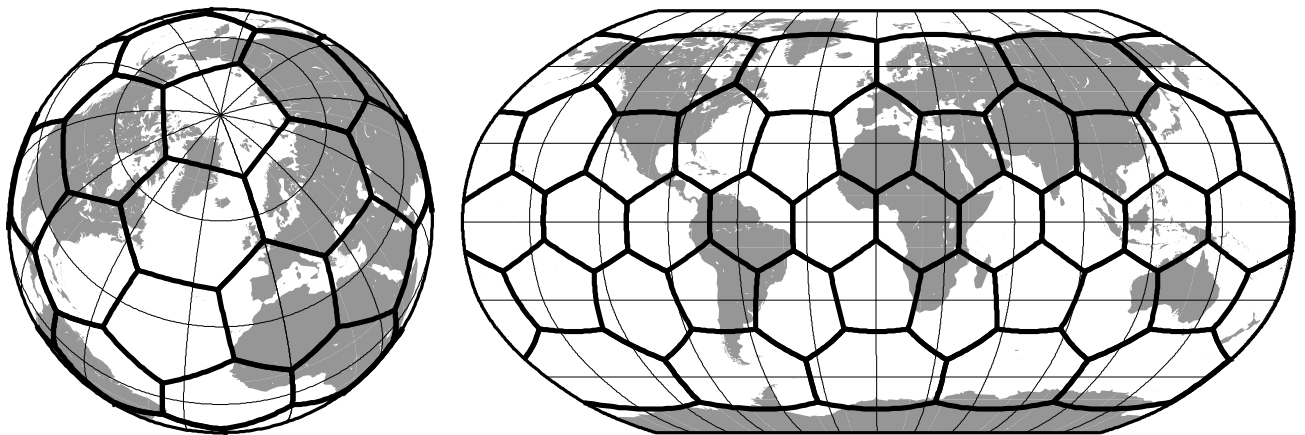


Figure B.2.: Orthographic and Robinson projection of the unstructured grid cells

# Operator index

## A

abs	57
acos	57
add	59
addc	58
asin	57
atan	57
atan2	59

## C

cat	25
chcode	46
chlevel	46
chlevelc	46
chlevelv	46
chname	46
const	124
copy	25
cos	57

## D

dayavg	84
daymax	84
daymean	84
daymin	84
daypctl	85
daystd	84
daysum	84
dayvar	84
delcode	31
delname	31
detrend	103
diff	20
diffv	20
div	59
dive	58
divdpm	62
divdpy	62
dv2uv	116
dv2uyl	116

## E

eca_cdd	131
eca_cfd	131
eca_csu	132
eca_cwd	132
eca_cwdi	133
eca_cwfi	133
eca_etr	134
eca_fd	134

eca_gsl	135
eca_hd	136
eca_hwdi	136
eca_hwfi	137
eca_id	137
eca_r10mm	138
eca_r20mm	138
eca_r75p	139
eca_r75ptot	139
eca_r90p	140
eca_r90ptot	140
eca_r95p	141
eca_r95ptot	141
eca_r99p	142
eca_r99ptot	142
eca_rr1	143
eca_rx1day	143
eca_rx5day	144
eca_sdii	144
eca_su	144
eca_tg10p	146
eca_tg90p	146
eca_tn10p	147
eca_tn90p	147
eca_tr	148
eca_tx10p	148
eca_tx90p	149
enlarge	53
ensavg	67
ensmax	67
ensmean	67
ensmin	67
enspctl	67
ensstd	67
enssum	67
ensvar	67
eq	40
eqc	41
exp	57
expr	56
exprf	56

## F

fdns	127
fldavg	69
fldmax	69
fldmean	69
fldmin	69
fldpctl	69
fldstd	69

fldsum .....	69
fldvar .....	69
<b>G</b>	
ge .....	40
gec .....	41
genbic .....	107
genbil .....	107
gencon .....	107
gendis .....	107
gp2sp .....	115
gp2spl .....	115
gradsdes1 .....	122
gradsdes2 .....	122
gridarea .....	121
griddes .....	23
gridweights .....	121
gt .....	40
gtc .....	41
<b>H</b>	
histcount .....	126
histfreq .....	126
histmean .....	126
histsum .....	126
houravg .....	82
hourmax .....	82
hourmean .....	82
hourmin .....	82
hourpctl .....	83
hourstd .....	82
hoursum .....	82
hourvar .....	82
hurr .....	128
<b>I</b>	
ifnotthen .....	37
ifnotthenc .....	38
ifthen .....	37
ifthenc .....	38
ifthenelse .....	37
info .....	18
infov .....	18
input .....	118
inputext .....	118
inputsrv .....	118
int .....	57
interpolate .....	109
intgridbil .....	109
intntime .....	112
inttime .....	112
intyear .....	113
invertlat .....	49
invertlatdata .....	49
invertlatdes .....	49
invertlon .....	49
invertlondata .....	49
invertlondes .....	49

<b>L</b>	
le .....	40
lec .....	41
ln .....	57
log10 .....	57
lt .....	40
ltc .....	41

<b>M</b>	
map .....	18
maskindexbox .....	51
masklonlatbox .....	51
maskregion .....	50
mastrfu .....	125
max .....	59
meravg .....	73
merge .....	26
mergetime .....	26
mermax .....	73
mermean .....	73
mermin .....	73
merpctl .....	73
merstd .....	73
mersum .....	73
mervar .....	73
min .....	59
ml2hl .....	111
ml2pl .....	111
monadd .....	60
monavg .....	86
monddiv .....	60
monmax .....	86
monmean .....	86
monmin .....	86
monmul .....	60
monpctl .....	87
monstd .....	86
monsub .....	60
monsum .....	86
monvar .....	86
mul .....	59
mulc .....	58
muldpm .....	62
muldpy .....	62

<b>N</b>	
ndate .....	21
ne .....	40
nec .....	41
nint .....	57
nlevel .....	21
nmon .....	21
npar .....	21
ntime .....	21
nyear .....	21

<b>O</b>	
output .....	119

outputtext .....	119
outputf .....	119
outputint .....	119
outputsrv .....	119

## P

pardes .....	23
--------------	----

## R

random .....	124
regres .....	103
remap .....	108
remapbic .....	106
remapbil .....	106
remapcon .....	106
remapdis .....	106
remapeta .....	110
replace .....	25
rotuvb .....	125
runavg .....	78
runmax .....	78
runmean .....	78
runmin .....	78
runpctl .....	79
runstd .....	78
runsum .....	78
runvar .....	78

## S

seasavg .....	90
seasmax .....	90
seasmean .....	90
seasmin .....	90
seaspctl .....	91
seasstd .....	90
seassum .....	90
seasvar .....	90
selcode .....	31
seldate .....	33
selday .....	33
selgrid .....	31
selgridname .....	31
selhour .....	33
selindexbox .....	35
sellevel .....	31
sellonlatbox .....	35
selltype .....	31
selmon .....	33
selname .....	31
selseas .....	33
selsmon .....	33
selstdname .....	31
seltabnum .....	31
selttime .....	33
seltimestep .....	33
selyear .....	33
selzaxis .....	31
selzaxisname .....	31

setcalendar .....	44
setcindexbox .....	52
setclonlatbox .....	52
setcode .....	43
setctomiss .....	54
setdate .....	44
setday .....	44
setgatt .....	48
setgatts .....	48
setgrid .....	47
setgridtype .....	47
setlevel .....	43
setltype .....	43
setmisstoc .....	54
setmissval .....	54
setmon .....	44
setname .....	43
setpartab .....	43
setreftime .....	44
setrtoc .....	123
setrtoc2 .....	123
setrtomiss .....	54
settaxis .....	44
settime .....	44
setunits .....	44
setyear .....	44
setzaxis .....	47
shifttime .....	44
showcode .....	22
showdate .....	22
showformat .....	22
showlevel .....	22
showltype .....	22
showmon .....	22
shownname .....	22
showstdname .....	22
showtime .....	22
showyear .....	22
sin .....	57
sinfo .....	19
sinfop .....	19
sinfov .....	19
smooth9 .....	123
sp2gp .....	115
sp2gpl .....	115
sp2sp .....	115
sputc .....	115
splitcode .....	27
splitday .....	28
splitgrid .....	27
splithour .....	28
splitlevel .....	27
splitmon .....	28
splitname .....	27
splitseas .....	28
splitsel .....	29
splityear .....	28
splitzaxis .....	27

sqr	57
sqrt	57
strbre	128
strgal	128
strwin	127
sub	59
subc	58
subtrend	104

**T**

tan	57
timavg	80
timmax	80
timmean	80
timmin	80
timpctl	81
timselavg	76
timselmax	76
timselmean	76
timselmin	76
timselpctl	77
timselstd	76
timselsum	76
timselvar	76
tmsort	124
timstd	80
tmsum	80
timvar	80
trend	104

**U**

uv2dv	116
uv2dvl	116

**V**

vct	23
vertavg	75
vertmax	75
vertmean	75
vertmin	75
vertstd	75
vertsum	75
vertvar	75

**W**

wct	127
-----	-----

**Y**

ydayavg	93
ydaymax	93
ydaymean	93
ydaymin	93
ydaypctl	94
ydaystd	93
ydaysum	93
ydayvar	93
ydrunavg	99
ydrunmax	99
ydrunmean	99

ydrunmin	99
ydrunpctl	101
ydrunstd	99
ydrunsum	99
ydrunvar	99
yearavg	88
yearmax	88
yearmean	88
yearmin	88
yearpctl	89
yearstd	88
yearsum	88
yearvar	88
yhouravg	92
yhourmax	92
yhourmean	92
yhourmin	92
yhourstd	92
yhoursum	92
yhourvar	92
ymonadd	61
ymonavg	95
ymondiv	61
ymonmax	95
ymonmean	95
ymonmin	95
ymonmul	61
ymonpctl	96
ymonstd	95
ymonsub	61
ymonsum	95
ymonvar	95
yseasavg	97
yseasmax	97
yseasmean	97
yseasmin	97
yseaspctl	98
yseasstd	97
yseassum	97
yseasvar	97

**Z**

zaxisdes	23
zonavg	71
zonmax	71
zonmean	71
zonmin	71
zonpctl	71
zonstd	71
zonsum	71
zonvar	71