A Subsetting Program for NCEP/NCAR Reanalysis netCDF Files: Release 2

Trent Technical Note 2008-3

J. Graham Cogley
October 2008
A Subsetting Program for NCEP/NCAR Reanalysis netCDF Files:
Release 2

Trent Technical Note 2008-3
J. Graham Cogley, October 2008

1 INTRODUCTION

1.1 NTFETCH

This report documents NTFETCH, a simple Windows console program for reading netCDF data files and extracting data subsets from them. Although the program is potentially quite versatile, it is tailored quite specifically to files containing meteorological data (geopotential heights, etc.) from the NCEP/NCAR Reanalysis (Kalnay et al. 1996). These files are obtainable from a number of locations. The ones used in developing and testing NTFETCH were from the NOAA-CIRES Climate Diagnostics Center, Boulder (http://www.cdc.noaa.gov/cdc/data.ncep.reanalysis.html).

NTFETCH is written in Fortran 95. The core of the program is a library of subroutines obtained from Unidata and documented by Rew et al. (1997) and Rew and Pincus (2002): http://www.unidata.ucar.edu/packages/netcdf/index.html. Because of difficulty in getting some of the Unidata Fortran 90 subroutines to work correctly, NTFETCH calls older Fortran 77 subroutines to access data. In fact both these sets of Fortran routines are simply wrappers around an inner core of routines written in C.

Above the core subroutines there is a layer of local subroutines which manage the handling of netCDF files: opening them; exploring and logging their contents; and writing selected subsets of the data to smaller output files in either flat-binary or plain-ASCII formats. Above the management layer is a layer of user-interface routines. The user sees a simple scripting language in which it is easy to describe subsets and request data operations. This language is documented in more detail below.

NTFETCH may be used freely, and its source code may be modified freely, provided that it not be sold for profit without the written agreement of the copyright holder and that its use or modification be duly acknowledged by citing this technical note.

1.2 SYNTAX

In the following documentation of command-prompt and control-file instructions,

- Literal text is printed in Roman font. Literal text, including apostrophes and quotation marks, is to be entered as it appears, except that upper- and lower-case forms of the same letter may be substituted for each other. For example “Jan” is to be entered as the letters J, a and n, although j,a,n and J,A,N would also be acceptable. In character strings enclosed by apostrophes, two consecutive apostrophes must be used to represent a single apostrophe. For example “Deadman”s Gulch” will be output as “Deadman's Gulch”.

- Variables are printed in italic font. A variable is a quantity for which some suitable text, such as a file name or a date, is to be supplied. “jan” is an example of a possible variable name.

- An ellipsis “...” indicates arbitrary, optional continuation text, as in “Jan ...”, in which the ellipsis means that “January” and “Jan” both satisfy the requirement.

- Vertical bars, “[ ]”, separate the items in a list of choices.

- Angle brackets, “< >”, enclose mandatory choices. That is, exactly one of the items within the brackets must be selected. (The angle brackets and bars themselves must not be typed.)

- Brackets, “[ ]”, enclose optional choices. That is, one of the items within the brackets may be selected, but none need be selected. (The brackets and bars themselves must not be typed.)
1.2.1 Time Words

When NTFETCH requires a date or a time, it relies on the notion of a time word, a word which
represents a quantity of some unit of time.

Time words are sequences of alphanumeric characters (".", that is, a dot; “0” to “9”; “A” to “Z”; “a” to “z”), separated by non-alphanumeric characters. Non-alphanumeric characters include blanks, equals signs and hyphens. Minus signs are not understood in this context, and plus signs are also non-alphanumeric.

A time word consists of a prefix and a suffix. If it begins with a letter, it contains no prefix. If it begins with a dot or a numeral, the prefix is the part of the word up to but excluding the first letter. The suffix is the part of the word from the first letter, if any, to the last character. The prefix and suffix may not be separated by non-alphanumeric characters.

Prefixes followed by an optional “AD...” or “BC...” suffix represent years. The suffix is not optional for years before 32AD.

Suffixes which have no prefix and are recognizable as the name in English of one of the twelve Gregorian months represent months. At least the first three letters must be supplied.

Prefixes which constitute a number between 1 and 31 and have no suffix represent days.

Prefixes followed by suffixes “h...”, “m...” and “s...” represent hours, minutes and seconds respectively.

1.3 INSTALLATION AND OPERATION OF NTFETCH

NTFETCH consists of a batch file, NTFETCH.BAT, and an executable file, NTMAIN.EXE. Source-code files and a Microsoft make file are also supplied for those who may need them.

Under the Windows Vista operating system the installation of NTFETCH simply requires copying the batch and executable files to a convenient directory. If you want to run NTFETCH from a different working directory, two further steps are necessary. First, a path to the installation directory should be specified as an operating-system environment variable, for example by opening a Windows command-prompt window (a “DOS box”) and typing

\[\text{path} \text{ desiredpath;}\%\text{PATH}\%\]

at the command prompt. Second, the line in NTFETCH.BAT which reads “ntmain” should be edited (in a text editor such as Notepad) by adding the path to the installation directory in front of the “ntmain”.

To run NTFETCH, open a command-prompt window and at the command prompt type

\[\text{ntfetch} \text{ filename}\]

where filename is the name of a control file containing script instructions. This control file must exist. If it is not in the current working directory, the file name must be preceded by its path name.

NTFETCH recognizes the script instructions described in section 2. Each instruction occupies one line of the control file. Blank lines, and lines whose first non-blank character is a colon, “:”, are ignored. More generally, NTFETCH ignores anything it cannot understand, which means that you can add ordinary English words to the instructions to make them more readable.

An NTFETCH run is complete when the last instruction in the control file has been read and processed. NTFETCH’s output consists of
1) zero or more data files containing subsetted data; in a successful run there is usually one output file per input data file;
2) a log file whose name is composed by prefixing “NT.” to the control file name, with the extension of the control file replaced by “.LOG”. The extension is everything including and following the last dot in the file name. If the control file has no extension, NTFETCH appends “.LOG” to form the log file name.
2 NTFETCH INSTRUCTIONS

2.1 SCRIPT INSTRUCTIONS

Instruction names in NTFETCH are not case-sensitive. They may be abbreviated to their first four letters. The available instructions are of three kinds. Environment descriptors describe the computing environment in which actions are carried out. They are:

- PATH
- INFIx
- FORMat
- HEADer
- CONServe
- ECHO
- TERSe
- VERBose
- CONVert

Data descriptors describe the location, size and spacing of subsets which are to be extracted from the input source. They are:

- YEAR
- DELTat
- SPAN
- SERIes
- TIMEsteps
- LONGitude
- LATItude
- LEVEL
- COLUmn
- POINt

Actions are instructions which implement the two kinds of thing NTFETCH knows how to do: exploring input files, and extracting subsets from them. They are:

- EXPLore
- FETCh
- OPEN
- APPEnd
- CLOSe

An Explore instruction yields an extended log of the contents of a netCDF file. A Fetch instruction yields nearly the same log, but also writes a subset of the data in the netCDF file to an output file. These two instructions rely on the current state of the environment descriptors and data descriptors. Explore instructions need only a Path instruction (at most). Fetch instructions need to know more; in particular, they assume explicitly that input will be taken from Reanalysis files, and rely on data-descriptor instructions when building input file names. There may be any number of Explore and Fetch instructions in a single control file, although it probably would not make sense to mingle the two. Open, Append and Close together form an extension of the Fetch instruction, in which the output file opened by the Open instruction remains open until closed by a Close, Open or Fetch instruction.

Each new action is governed by the values specified in the most recent instance of each environment or data descriptor.
2.2 Environment Descriptors

2.2.1 PATH
The Path instruction takes the form

    Path [input | output] ‘path’

where path is a character string which specifies a path to files. Note the required enclosing apostrophes. If “input” or “output” is present, the path is read as being for input files or output files respectively. If neither word is present, the specified path is the same for both input and output files, and output files will be in the same directory as input files.

If no Path instruction is given, input and output will be from and to the current working directory.

Example
    Path Input 'c:\ncepncar\'
    Path Output ' '

specifies an input directory, with output going to the current working directory. The terminal backslash should be present in path, but NTFETCH supplies it if it is omitted.

2.2.2 INFIX
The Infix instruction gives information about the structure of input file names, and takes the form

    Infix 'infixtext'

where infixtext is a character string to be inserted in the middle of input file names. See section 3.1 for an explanation of how NTFETCH composes the names of the files it needs to read.

If infixtext begins or ends with a dot, the dot will be removed.

Example
    Infix 'eatm'

specifies that the string “eatm” (which signifies “entire atmosphere” in the Reanalysis naming system) is to be inserted in the appropriate place in input and output file names.
2.2.3 FORMAT

The Format instruction is

\[
\text{Format} < \text{fmt}' | \text{binary}> 
\]

where \text{fmt} is a Fortran format specification to be used when writing data subsets.

The type of the input data is described in the log generated by the Explore and Fetch instructions. To match the output format to the input type, you need to understand netCDF’s data types, which are explained in section 3.2. Data of type Character are not understood by this version of \text{NTFETCH}. To convert integer input to real output or real input to integer output, simply specify the appropriate integer or real output format.

“Format binary” will cause output files to be opened as flat binary files, to which any subset of \text{N} elements will be written as \text{N} binary numbers. In this case, the type of the output is determined by \text{NTFETCH} from information in the input file. The types of the variable to be fetched and of two of its netCDF attributes, \text{scale_factor} and \text{add_offset}, are compared; the highest of these three types is chosen for the binary output, and is reported in the log. See the Fetch instruction (section 2.4.2) for more information on \text{scale_factor} and \text{add_offset}.

Examples

Format ‘(221i6)’
will cause any subset to be written out as 221 integers per line, each integer occupying 6 characters; while

Format ‘(8f9.2)’
gives 8 reals (floating-point numbers) per line, each real occupying 9 characters of which two will be decimal digits and one will be the decimal point.

If the field width (221 and 8 in the two examples just given) is not a divisor without remainder of the number of integers or reals to be output, the final line of output will be short. Some programs may be confused by this when they attempt to read the output file.

Notice that Fortran format specifications must begin and end with parentheses. (The enclosing apostrophes are required by \text{NTFETCH}, not by Fortran.)

Although you can probably get the output you need with one of the two kinds of specification just illustrated (“i” and “f”), some other Fortran format specifications can also be tried. For example

Format ‘(d18.7)’
and

Format ‘(e18.7)’
will output a subset as one element per line in double-precision and single-precision scientific notation respectively, while

Format ‘(10i6.4)’
will output 10 integers per line in fields 6 characters wide, with not fewer than 4 digits per integer (including leading zeros if necessary).
2.2.4 HEADER
The Header instruction is

    Header ['headertext']

where headertext is a character string which will be written as the first record of any subsequently opened output data file if “Format binary” has not been specified.

Examples
    Header ’Mean monthly air temperature, Mexico, 1961-1990’
    Header ‘’
will produce a header text and a blank header line respectively.

2.2.5 CONSERVE
The Conserve instruction is

    Conserve < speed | memory >

where “Conserve speed” is the default. “Conserve memory” causes NTFETCH to try to use less memory when extracting subsets.

Ordinarily an entire subset is read into memory from the current input file and then immediately written out. This is fast, but may lead to trouble if the subset is large by comparison with the available memory. When this happens, conserving memory might help. NTFETCH will examine the attributes of the specified variable in search of its record dimension, which is its most slowly varying non-trivial dimension. “Non-trivial” means having a length greater than one element, while “most slowly varying” refers to the order in which the elements of the dataset are stored (see section 3.3). When conserving memory, NTFETCH will read and write only one slice at a time from this dimension rather than reading and writing the whole subset at once. This can reduce the memory requirement dramatically.

Usually, but not necessarily, the most slowly varying dimension is time.

2.2.6 ECHO
The Echo instruction takes the form

    Echo < off | on >

where “off” is the default and “on” causes NTFETCH to report on instructions as it finds them in the control file. The report is sent to standard output. This facility can be useful in diagnosing the cause of unexpected results if they arise from incorrect instruction syntax.

2.2.7 TERSE
The Terse instruction is

    Terse [no]

It reduces the quantity of information written to the log file, except that when “no” is present the instruction means the same as “Verbose” (section 2.2.8). The default behaviour is “Verbose.”
2.2.8 VERBOSE
The Verbose instruction is

Verbose [no]

It causes all available information to be written to the log file, which is the default behaviour. “Verbose no” means the same as “Terse” (section 2.2.7).

2.2.9 CONVERT
The Convert instruction takes the form

Convert [gainntag] [offset ntoffset]

wherentag is a scaling factor and ntoffset an addend for output numbers.ntag must not be zero.

Ordinarily the numbers output by NTFETCH are defined by

\[ v_{out} = scale\_factor \times v_{in} + add\_offset , \]

where scale_factor and add_offset are attributes of the variable v_in obtained from the input file (section 2.4.2). Whenntag and ntoffset are specified, the numbers output are

\[ v_{out} = (ntag \times scale\_factor) \times v_{in} + (ntag \times add\_offset + ntoffset) . \]

That is, the stored number v_in is first converted as specified in the input file, and the result is further converted as specified in the Convert instruction.

Examples

Convert gain = 1.8 ; offset = 32
will convert input temperatures in degrees Celsius to output temperatures in degrees Fahrenheit.

Note the punctuating “=” and “;” in these examples, which NTFETCH ignores.
2.3 Data Descriptors
Most of the data-describing instructions specify the start and end of a subset along one of its dimensions, together with the interval between subset samples.

If any of the instructions corresponding to data dimensions (Longitude; Latitude; Level; Column or Point; Span, Series or Timesteps) are omitted, NTFETCH will output, from each input file read, the entire extent of the dataset along that dimension. The Year instruction, however, may only be omitted if there is a Series instruction which specifies a start year and end year. If a specified start–end range overlaps the available extent of the dataset, the output will be truncated. A specified start–end range which is disjunct from the extent of the dataset, starting beyond the end of the extent or ending beyond the start, will yield no output.

NTFETCH can explore netCDF datasets having up to seven dimensions but can only read and write variables of up to four dimensions, each of which must be recognizable as belonging to the set (longitude, latitude, level, time).

2.3.1 YEAR
The Year instruction is

\[
\text{Year} < \text{startyear} [\text{endyear} [\text{stride}]] | \text{none} >
\]

where \text{startyear} and \text{endyear} are integers representing the first and last years in a sequence of years from which data are to be extracted and \text{stride} is the integer number of years between subset years. \text{endyear} must not be earlier than \text{startyear}. If omitted it is assumed to be equal to \text{startyear}. If omitted, \text{stride} is assumed to be equal to 1. To omit \text{startyear} is an error.

“Year none” specifies that the input file contains time-invariant data or data not organized by year. Do not leave out the Year instruction altogether in this case, for without it NTFETCH may be unable to construct the file names it needs.

When a Span instruction (section 2.3.3) is given, a Year instruction must accompany it.

Examples

\[
\text{Year 1998}
\]

specifies that there is to be a single sample year, 1998, while

\[
\text{Year 1998 2004 2}
\]

2.3.2 DELTAT
The Deltat instruction has the form

\[ \text{Deltat} < \text{interval} \mid \text{adverb} > \]

where \textit{interval} is a single time word representing days or shorter units and defining the interval between elements along the time dimension of the input dataset. \textit{adverb} is an adverb describing this interval.

\textit{interval} is converted to hours. If you need to specify an \textit{interval} longer than 31 days you must do so in hours or shorter units. At present the only \textit{adverb} recognized by NTFETCH is “monthly”.

Although Deltat can be obtained from the input files which contain the variable data, NTFETCH may need to know it before opening any input files. To be precise, Deltat is useful, and may be essential, when the Span or Series instructions (sections 2.3.3, 2.3.4) are used. It must appear before either of these instructions, for it is the means by which they locate the last timestep of a month or day when their start and end dates do not give the time of day explicitly. In other cases, Deltat is not needed.

Examples

Deltat 6h
says that the input file contains 6-hourly data, while

Deltat 24hours
Deltat 1.0
Deltat 1 day

all specify daily data. Notice that “24hours” is a time word with prefix and suffix (section 1.2.1). However, “1 day” is not. In the latter, the “day”, with its preceding blank, is redundant but harmless, and the time word is “1”. In the former, if there were a blank between “24” and “hours” NTFETCH would conclude that Deltat should be set to 24 days.

Deltat monthly
says that the interval along the time dimension is 1 month.
2.3.3 SPAN
The Span instruction is

    Span [start *startdate*] [end *enddate*] [every *stride*]

where *startdate* and *enddate* are calendar dates (civil dates in the Gregorian calendar), possibly including times of day; dates are sequences of time words. *stride* is the positive integer interval between samples along the time dimension, in numbers of steps.

You can include a year in either of the dates, but it will be ignored. The Span instruction assumes that years are given by the Year instruction (section 2.3.1), which must not be omitted. A different output file is produced for each year specified by the Year instruction, containing data for the span specified by the Span instruction.

If *startdate* is omitted, the span is assumed to begin at the first timestep of the current year. If *enddate* is omitted, the span is assumed to end at the last timestep of the current year. If *stride* is omitted it is assumed to be equal to 1.

The format for dates is flexible (section 1.2.1). In particular, the different time words may appear in any order and may be separated by any one or more non-alphanumeric characters.

Dates need not be complete, but must be unambiguous in their context. Start and end dates which consist only of month names are interpreted respectively as identifying the first or last timestep of the month in question. Similarly, dates which consist of a month and a day identify the first or last timestep of that day of that month. Hours and smaller subdivisions of time, when encountered, are converted to fractional days. Thus, when Deltat (section 2.3.2) is 6 hours, the end date “April” refers to the fourth timestep of 30 April (at 18h); the start date “30 April” refers to the first timestep of 30 April (at 00h); and “12.00h 30 April” refers to the third timestep of 30 April (at 12h) whether it is a start date or an end date.

If *startdate* is later than *enddate* the Span instruction is interpreted as specifying a time span which extends beyond the end of the current year into the next year.

NTFETCH understands leap years. You can avoid having to describe the length of each February explicitly by giving the end date “February”.

If the Span instruction is omitted, and there is no Series or Timesteps instruction either (sections 2.3.4, 2.3.5), the entire time dimension of each year indicated by the Year instruction is selected for output and is written to a different file for each year. That is, Span is the default instruction.

Examples

Span start 1 July end 31 August
reads data for every timestep from the first timestep of 1 July to the last timestep of 31 August.

Span start Dec end Feb every 4
reads data for every fourth timestep from the first timestep of December to the last timestep of the following February.

Year 2003 2004
Span start 27 February end 01 March
produces three days of output for 2003 and four for the leap year 2004.

Year 2003 2004
Span start 27 February end February
produces two days of output for 2003 and three for 2004. You cannot say “end 29 February” because the Span instruction rejects “29 February”. You can say “end 28 February”, which will yield two days of output for each year.
2.3.4 SERIES

The Series instruction is

    Series start startdate end enddate [every stride]

where startdate and enddate are calendar dates (civil dates in the Gregorian calendar), possibly including times of day; dates are sequences of time words. stride is the positive integer interval between samples along the time dimension, in numbers of steps. If it is omitted it is assumed to be equal to 1.

The Series instruction generates a single continuous series extending from the start date to the end date, neither of which may be omitted. The end date must not be earlier than the start date, but it can be in the same year or a later year. The series is written to a single output file identified with the year of the start date.

The format for dates is flexible (section 1.2.1). In particular, the different time words may appear in any order and may be separated by any one or more non-alphanumeric characters.

Dates need not be complete, but must be unambiguous in their context. They must contain a month, and may contain a year. If the year is omitted it will be supplied from the Year instruction (section 2.3.1). Start and end dates which consist only of month names are interpreted respectively as identifying the first or last timestep of the month in question. Similarly, dates which contain a month and a day identify the first or last timestep of that day of that month. Hours and smaller subdivisions of time, when encountered, are converted to fractional days, which are then used to identify the timestep. Thus, when Deltat (section 2.3.2) is 6 hours, the end date “April” refers to the fourth timestep of 30 April (at 18h); the start date “30 April” refers to the first timestep of 30 April (at 00h); and “12.00h 30 April” refers to the third timestep of 30 April (at 12h) whether it is a start date or an end date.

NTFETCH understands leap years. You can avoid having to describe the length of each February explicitly by giving the end date “February”.

If the Series instruction is omitted, and there is no Span or Timesteps instruction either (sections 2.3.3, 2.3.5), the entire time dimension of each year indicated by the Year instruction is selected for output and is written to a different file for each year. That is, a set of year-long spans is extracted, not a multi-year series.

Examples

Series start 2000 September 1 end 2002 August 31
reads data for every timestep from the first timestep of 1 September 2000 to the last timestep of 31 August 2002.

Series start Dec end Feb every 4
reads data for every fourth timestep from the first timestep of December in the startyear specified in the Year instruction to the last timestep of February in the endyear specified in the Year instruction.

Series start Jan 1990 end Dec 1999
reads data for every timestep from the first of 1990 to the last of 1999.
2.3.5 TIMESTEPS
The Timesteps instruction is

\[
\text{Timesteps \ startstep \ [\ endstep \ stride]}\]

where \(\text{startstep}\) and \(\text{endstep}\) are the positive integer indices of the first and last elements to be sampled along the time dimension of the input dataset, and \(\text{stride}\) is the positive integer interval between each sample along the time dimension, in numbers of steps. If \(\text{endstep}\) is omitted, the selected subset extends from \(\text{startstep}\) to the last timestep of each year specified in the Year instruction (section 2.3.1). If \(\text{stride}\) is omitted it is assumed to be equal to 1. The Timesteps instruction thus takes units of integer timesteps for all three of its arguments.

\(\text{startstep}\) greater than \(\text{endstep}\) is an error.

If the Timesteps instruction is omitted, and there is no Span or Series instruction either (sections 2.3.3, 2.3.4), the entire time dimension of each year indicated by the Year instruction is selected for output and is written to a different file for each year.

Examples

Timestep 121
specifies that all time slices are to be extracted from the 121st to the last of each year specified in the Year instruction.

Timestep 121 121
specifies that the 121st time slice is to be taken from the input file for each year.

Timestep 121 124 2
specifies that the 121st and 123rd time slices are to be taken from the input file for each year.

Note the spelling “Timestep”. NTFETCH recognizes only the first four of the letters of each instruction, in upper or lower case, so that “Time”, “TIMESTEP” and “Timesteps” are equivalent.
### 2.3.6 LONGITUDE

The Longitude instruction is

```
Longitude startlon [endlon [stride]]
```

where `startlon` and `endlon` are the first and last longitudes to be sampled along the longitude dimension of the input dataset, and `stride` is the positive integer interval between each sample along the longitude dimension, in numbers of steps. If `endlon` is omitted the selected subset extends from `startlon` to the end of the longitude dimension (just west of the Greenwich meridian). If `stride` is omitted it is assumed to be equal to 1.

`startlon` and `endlon` have units of degrees east of the Greenwich meridian. Any longitudes between -180° and 0° are added to 360° to bring them into the range [180° ... 360°). After this correction, longitudes outside the range [0° ... 360°) are treated as errors.

If `startlon` is greater than `endlon`, NTFETCH assumes that the subset to be extracted straddles the Greenwich meridian.

If the Column or Point instruction is given, the results of any preceding instance of the Longitude instruction are overridden. If the Longitude instruction is omitted, the entire longitude dimension is selected for output.

**Examples**

Longitude 265.0 300.0 1

and

Longitude -95.0 -60.0 1

both specify a subset containing every data element along the longitude dimension between -95° and -60° (95° West to 60° West) inclusive. The step size in degrees, which happens to be 2.5° in Reanalysis files, is not needed. Assuming that it is indeed 2.5°, then

Longitude 340.0 12.5 3

extracts and writes out data elements at the East longitudes 340.0°, 347.5°, 355.0°, 2.5°, 10.0°, in that order.

Longitude 180.0 177.5

converts from the Greenwich-to-Greenwich format, as in Reanalysis files, to the common alternative dateline-to-dateline format.
2.3.7 LATITUDE

The Latitude instruction is

Latitudate startlat [endlat [stride]]

where startlat and endlat are the first and last latitudes to be sampled along the latitude dimension of the input dataset, and stride is the positive integer interval between each sample along the latitude dimension, in numbers of steps. If endlat is omitted the selected subset extends from startlat to the end of the latitude dimension (the southernmost latitude). If stride is omitted it is assumed to be equal to 1.

startlat and endlat have units of degrees of latitude. Latitudes must lie in the range from -90° at the South Pole to 90° at the North Pole.

If the Column or Point instruction is given, the results of any preceding instance of the Latitude instruction are overridden. If the Latitude instruction is omitted, the entire latitude dimension is selected for output.

Examples

Latitude 60.0 32.5 1
specifies a subset containing every data element along the latitude dimension between 60.0° and 32.5° inclusive. In Reanalysis files the latitude typically decreases along the latitude dimension (that is, the North Pole comes first and the South Pole last), but the output of NTFETCH will be in whatever order is indicated by startlat and endlat. Thus

Latitude 32.5 60.0 1
will produce the same output information as the previous example, but the elements will be written in order of increasing latitude (south first, north last).
2.3.8 LEVEL
The Level instruction is

```
Level startlev [endlev [stride]]
```

where `startlev` and `endlev` are the first and last levels to be sampled along the vertical (pressure) dimension of the input dataset, in millibars, and `stride` is the positive integer interval between each sample along the vertical dimension, in numbers of levels. If `endlev` is omitted the selected subset extends from `startlev` to the end of the vertical dimension (the topmost level). If `stride` is omitted it is assumed to be equal to 1.

In Reanalysis files the vertical dimension illustrates the fact that steps not only need not be increasing but need not be uniform either. The first and last points along the vertical dimension are typically at 1000 mbar and 10 mbar, and those in between are separated by varying differences of pressure. NT FETCH recognizes this irregularity. It does, however, assume that coordinates vary monotonically, as in the sequence (1000, 900, 850, 800, 750). That is, changes of direction, as in (1000, 800, 900, 700, 600), would not be understood.

Data are written out in the order indicated by the start and end levels.

If the Point instruction is given, the results of any preceding instance of the Level instruction are overridden. If the Level instruction is omitted the entire vertical dimension is selected for output.

Examples

- Level 500
  specifies that all levels from 500 mbar to the top of the dataset are to be extracted, while
  Level 500 500
  specifies that only the 500-millibar level is to be extracted.

- Level 1000 400 2
  specifies every other level between 1000 mbar and 400 mbar inclusive.

- Level 400 1000 2
  will produce the same output information as the previous example, but the elements will be written in order of increasing pressure (top first, bottom last).

- Level 200 50
  will nominally sample all levels between 200 mbar and 50 mbar inclusive, but if, as in some files, the highest level is 300 mbar, there will be no output.
2.3.9 COLUMN
The Column instruction is

Column lon lat

where *lon* is a longitude and *lat* is a latitude. *lon* has units of degrees east of the Greenwich meridian. Longitudes between -180° and 0° are added to 360° to bring them into the range [180° ... 360°]. After this correction, longitudes outside the range [0° ... 360°) are treated as errors. *lat* has units of degrees of latitude. It must lie in the range from -90° at the South Pole to 90° at the North Pole.

Reanalysis columns which enclose the point (*lon*, *lat*) will be read, their extent in the vertical dimension being determined by the most recently given Level instruction (section 2.3.8). If the Level instruction is omitted the entire vertical dimension will be selected. The input data will be interpolated bilinearly to (*lon*, *lat*) on each level. The resulting output will be an interpolated column of numbers for each currently-specified time, and equivalent columns of geopotential heights written to a separate output file as described in section 2.4.2.

The Column instruction overrides any preceding instance, and is overridden by any following instance, of the Longitude and Latitude instructions. Strange results will be obtained if one of the latter instructions is given without the other after a Column instruction.

**Example**

```
Level     1000 300
Column -134.330 58.400
```
specifies that data will be interpolated above a point in southeast Alaska for levels between 1000 mbar and 300 mbar.

2.3.10 POINT
The Point instruction is

Point lon lat elev

where *lon* is a longitude, *lat* is a latitude and *elev* is an elevation (strictly, a geopotential height) in metres above sea level. *lon* has units of degrees east of the Greenwich meridian. Longitudes between -180° and 0° are added to 360° to bring them into the range [180° ... 360°). After this correction, longitudes outside the range [0° ... 360°) are treated as errors. *lat* has units of degrees of latitude. It must lie in the range from -90° at the South Pole to 90° at the North Pole.

All three coordinates are required, but if *elev* is omitted it is assumed to be 0 m.

Reanalysis columns which enclose the point (*lon*, *lat*) will be read. The entire vertical dimension is selected. The input data will be interpolated bilinearly to (*lon*, *lat*) on each level, and then linearly to the geopotential height *elev*. The resulting output will be a single interpolated number for each currently-specified time.

The Point instruction overrides any preceding instance, and is overridden by any following instance, of the Longitude, Latitude and Level instructions. Strange results will be obtained if one of the latter instructions is given without the others after a Point instruction.

**Example**

```
Point  -134.330 58.400  990
```
specifies that data will be interpolated to the elevation 990 m a.s.l. at a point in southeast Alaska.
2.4 Actions

2.4.1 EXPLORE
The Explore instruction has the form

   Explore 'filename'

and causes NTFETCH to desist from reading instructions in order to read the file filename and prepare a log of its contents. When searching for the file, it prefixes the current input path, if any, to filename. The log is written to the log file, which is in the current working directory (not the directory specified by the current output path). There is no provision for suppressing this log, but it can be abbreviated with the Terse instruction (section 2.2.7). See section 4.1 for a generalized explanation of the log.

   When the log is complete, NTFETCH closes the input file and returns to reading instructions from the control file.

   The Explore instruction does not make assumptions about the context of the netCDF file which is to be read. It can therefore be used to log the contents of any netCDF file, not just NCEP/NCAR Reanalysis files.

Example
   Explore 'hgt.1998.nc'
writes a log of the file HGT.1998.NC to the log file.
2.4.2 FETCH
The Fetch instruction has the form

    Fetch varname ['suffix'] ['prefixtext']

and causes NTFETCH to desist from reading instructions in order to read one or more files containing data
for the variable varname. varname must match exactly the name of the variable as it appears in the input
file names and as it is defined within the files. suffix, if present, is a string of characters which will be
used in composing the output file name; if it begins or ends with a dot, the dot is removed. prefixtext, if
present, is a string of characters which will be written at the beginning of each output record.

    Input file names are composed as described in section 3.1, and the current input path, if any, is
prefix to each name.

    Fetch closes any output files which are open, and composes a new output file name as described
in section 3.1. The suffix is inserted in the output file name just before the extension, and thus serves to
distinguish between output subsets which would otherwise have conflicting file names. (Without a
suffix, each new Fetch instruction would overwrite the file or files created for variable varname by its
predecessor.) The current output path, if any, is prefixed to the composed file name, and the file is
opened.

    The data are extracted from each input file according to whatever start, end and stride are current
along each dimension of the dataset, up to a maximum of four dimensions. They are then multiplied by
the scale_factor specified as an attribute of the variable in the current input file, and the add_offset
specified in the input file is added. (The values of these two quantities can be found in the log file, where
they are listed as Gain and Offset respectively; see section 4.1.2.1.) The result of this operation may be
further modified if the Convert instruction (section 2.2.9) has been given.

    The data are then written to the output file. When Fetch writes data in accordance with the
Column instruction, a second output file is also written. It contains the geopotential heights corresponding
to the levels on which data have been interpolated, and has the same name as the current output file but
with varname replaced by “hgt”.

    If a prefixtext is supplied, and the format specified in the Format instruction is not binary, the
prefixtext is written (with both leading and trailing blanks being preserved) at the beginning of each
output record. Note that a record (i.e. the entire contents of a subset) is not in general the same thing as a
line. For example, the result may be undesirable if the Format instruction specifies a number of numbers
per line of output which is less than the number of numbers per output record: when a record containing
10 years of monthly output is written with a format specifying only 12 numbers per line, only the first line
will have the prefixtext.

    Each Fetch instruction generates a log of the contents of the file or files which contain varname.
The log is substantially the same as that generated by the Explore instruction (section 2.4.1). It is written
to the log file in the current working directory (not the directory specified by the current output path).
There is no provision for suppressing this log, but it can be abbreviated with the Terse instruction (section
2.2.7). See section 4.1 for a generalized explanation of the log.

    Once the output file or files have been written, they are closed and NTFETCH returns to reading
instructions from the control file.

Examples

    Fetch hgt
will read one or more files each containing one year’s worth of the variable named hgt (geopotential
height) and will write a subset file of geopotential heights for each year.
Year 1998 1999
Fetch hgt

Year 1998
Longitude  242.5  275.0
Latitude  35.0  12.5
Fetch hgt 'Mexico'
Longitude  15.0  35.0
Latitude  -20.0  -37.5
Fetch hgt 'S.Africa'
will create output files HGT.1998.MEXICO.OUT and HGT.1998.S.AFRICA.OUT containing subsets encompassing Mexico and South Africa respectively.
2.4.3 OPEN
The Open instruction has the form

    Open varname ['suffix '] ['prefixtext']

and is the same as the Fetch instruction (section 2.4.2) except that the output file or files remain open after processing is complete. Output generated subsequently by Append instructions, of which there may be any number, will be written to the already-open file or files.

Once opened by the Open instruction, files remain open until closed by a Fetch, Open or Close instruction.

2.4.4 APPEND
The Append instruction has the form

    Append varname ['prefixtext']

and writes data to an output file previously opened with the Open instruction (section 2.4.3). The data identified by varname are fetched, as described in section 2.4.2, from the appropriate input file, which need not be the same as that specified in the Open instruction.

Any number of Append instructions may be given while an output file remains open, but it is an error to give an Append instruction when no output file is open. That is, Append instructions may appear only between an Open instruction and a Fetch, Open or Close instruction.

The output file or files remain open after the Append instruction has completed.

Example

Infix   'mon.ltm'
...
Column  -134.330  58.400
Open    air    'TofP'    'Lemon Creek Gl'
Column  -90.830  79.500
Append  air    'White Gl'
Column  18.570  67.900
Append  air    'Storglaciaren'
Close

will open the file AIR.MON.LTM.NC to read data surrounding three glaciers (in Alaska, Arctic Canada and Sweden respectively). At each glacier a column of air temperatures is interpolated and written to the output file AIR.MON.LTM.TOFP.OUT, each record being prefixed by the name of the glacier. The corresponding geopotential heights are written to the file HGT.MON.LTM.TOFP.OUT.

2.4.5 CLOSE
The Close instruction is

    Close

and closes any output files which remain open after an Open or Append instruction has been processed.
3 NETCDF AND NCEP/NCAR REANALYSIS CONVENTIONS

3.1 NCEP/NCAR REANALYSIS FILE NAMES AND NTFETCH FILE NAMES
NCEP/NCAR Reanalysis file names are built up from the variable name varname, the infix text infixtext, the year number year and the standard extension "NC", according to the following file name template:

\[ \text{varname[.infixtext[.year].nc} \]

Consider the following examples:

<table>
<thead>
<tr>
<th>File name template</th>
<th>varname</th>
<th>infixtext</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAND.NC</td>
<td>land</td>
<td></td>
</tr>
<tr>
<td>HGT.year.NC</td>
<td>hgt</td>
<td></td>
</tr>
<tr>
<td>TMP.300CM.year.NC</td>
<td>tmp</td>
<td>300cm</td>
</tr>
<tr>
<td>PRES.SFC.year.NC</td>
<td>pres</td>
<td>sfc</td>
</tr>
<tr>
<td>RHUM.SIG995.year.NC</td>
<td>rhum</td>
<td>sig995</td>
</tr>
<tr>
<td>PRES.TROPP.year.NC</td>
<td>pres</td>
<td>tropp</td>
</tr>
<tr>
<td>DSWRF.NTAT.GAUSS.year.NC</td>
<td>dswrf</td>
<td>ntat.gauss</td>
</tr>
<tr>
<td>HGT.MON.LTM.NC</td>
<td>hgt</td>
<td>mon.ltm</td>
</tr>
</tbody>
</table>

The varname, always present, is specified in the Fetch instruction (section 2.4.2). The infixtext may or may not be present. If present, it is specified in the Infix instruction (section 2.2.2). It may describe the level of the data in the file, or it may contain other information. Whatever the content, NTFETCH needs to be told the infixtext explicitly.

The sequence of years from which the variable year is drawn is specified in the Year instruction (section 2.3.1) or possibly the Series instruction (section 2.3.4). File names for time-invariant data (for example those in LAND.NC), or data not organized into single-year files (as in HGT.MON.LTM.NC, a file of monthly long-term means), can be composed with the “Year none” variant of the Year instruction.

NTFETCH’s output files have names the same as their corresponding input files, with the possible addition of a suffix and with the extension “.NC” replaced by “.OUT”:

\[ \text{varname[.infixtext[.year][.suffix].out} \]

3.2 NETCDF DATA TYPES
The definition of netCDF (Rew and Pincus 2002) provides for a number of data types, one of which must be an attribute of any variable or a property of any attribute in a netCDF file. The data types, from lowest to highest, are

<table>
<thead>
<tr>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer1</td>
<td>One-byte integers</td>
</tr>
<tr>
<td>Character</td>
<td>One-byte characters (&quot;text&quot;)</td>
</tr>
<tr>
<td>Integer2</td>
<td>Two-byte integers</td>
</tr>
<tr>
<td>Integer</td>
<td>Four-byte integers</td>
</tr>
<tr>
<td>Real4</td>
<td>Four-byte real (floating-point) numbers</td>
</tr>
<tr>
<td>Real8</td>
<td>Eight-byte real numbers (&quot;double-precision&quot;)</td>
</tr>
</tbody>
</table>

Data of type Character are not understood by this version of NTFETCH. Although the storage size and precision of the numbers differ greatly between the different types, the main practical distinction when
preparing an NTFETCH control file is between the three integer types and the two real types. You can convert integers to reals and reals to integers with the Format instruction (section 2.2.3), but it is important to realize that such conversions can sacrifice numerical precision in the output of NTFETCH. You should also be aware that a variable whose type is integer may have a gain and offset of type real. This is commonly done to reduce (e.g. to only two bytes) the storage size of numbers that are intrinsically real but are not known with four or eight bytes of real precision.

### 3.3 COORDINATE VARIABLES AND RECORD VARIABLES

A netCDF file typically contains several “coordinate variables” and zero or more “record variables”. A coordinate variable is a one-dimensional variable whose name is the same as that of a dimension defined within the netCDF file. Its elements have values representing positions along the extent of the dimension. Each defined dimension has a name and a length, the latter being the number of elements in the dimension. NTFETCH lists all the coordinate variables whenever it generates a log.

For example, a typical Reanalysis file contains the coordinate variables `lon`, `lat`, `level` and `time`. `lon` and `lat` have lengths of 144 and 73 respectively, reflecting the 2.5° resolution of the Reanalysis. The variable `lon` extends from 0.0° to 357.5° of longitude, while the latitude variable `lat` extends from the North Pole to the South Pole inclusive. The vertical variable `level` generally has 17 irregularly-spaced elements between 1000 mbar and 10 mbar inclusive. The coordinate variable `time` contains the elements for the record dimension `time`.

For present purposes the record dimension may be defined as the dimension whose index values vary most slowly in the ordering of record-variable elements in memory. Record variables are arrays extending over one or more (usually more) of the defined dimensions. They are deemed by NTFETCH to be those variables whose names are not also dimension names. (This is a slightly different definition from that given by Rew and Pincus 2002.) The Fetch instruction (section 2.4.2) is designed to extract subsets from netCDF record-variable arrays.

For example, `HGT.1998.NC` is a Reanalysis file containing, in addition to the coordinate variables `lon`, `lat`, `level` and `time`, the record variable `hgt`, which is the geopotential height. Actually there are two files of that name, one for 6-hourly data and one for daily data. The former contains 1460 and the latter 365 elements, or “records”, along the time dimension. Each record represents the geopotential height field as an array of numbers, one at each of 144 longitudes, 73 latitudes and 17 levels.

The shape of an array is given by a list of dimension lengths, and its size is the product of those lengths. The shape of `hgt` in the daily `HGT.1998.NC` is therefore (144, 73, 17, 365), and its size is $144 \times 73 \times 17 \times 365 = 65,226,960$. 

---

22
4 THE NTFETCH LOG

4.1 CONTENTS OF THE NTFETCH LOG

The log file generated during each run of NTFETCH consists of introductory and descriptive sections for each input file. The log described below is that produced in the default Verbose mode. Several items are omitted in Terse mode.

4.1.1 Introductory Section

Introductory sections include information from the control file about the sample structure and output, if any, which have been specified in instructions preceding each Fetch instruction. The first introductory section begins with a version identifier and timestamp.

4.1.2 Descriptive Sections

The remaining sections describe the properties of the data in each input file and, for Fetch instructions, the shape of the subset to be extracted and the outcome of the fetch operation.

4.1.2.1 Input Data Properties

This section, generally the longest, is identical whether the generating instruction is Explore or Fetch.

The first information given is a list of the number of dimensions, variables, attributes of variables and global attributes defined in the input file. The index of the record dimension is also given; if there is no such dimension, the index is reported as -1. The record dimension, also called the unlimited dimension, is a netCDF concept (Rew and Pincus 2002) which is explained in section 3.3.

Next, for each global attribute the dimension index, length (either number of characters or number of array elements), data type, attribute name and attribute value are reported.

The name and length of each dimension are reported. The length is the number of data elements within the extent of the dimension.

For each variable in the input file, a substantial list of attributes is given. The list includes the variable name and also the:

- **Type** data type of variable elements (section 3.2)
- **Dimensions** number and indices of dimensions of variable
- **Attributes** same information as for global attributes above
- **Longname** descriptive name for variable
- **Units** physical units of stored data
- **Range** actual range of data in input file
- **RangeMax** maximum permitted range of data
- **Missing** data value representing missing information
- **Offset** \( \text{add}_\text{offset} \), value to be added when extracting stored number
- **Gain** \( \text{scale}_\text{factor} \), multiplier for stored number (before adding Offset)
- **OutType** highest among Type, type of Offset and type of Gain (section 3.2)

The data elements stored in the input file for each coordinate variable (section 3.3) are listed. When the length of the coordinate variable’s dimension exceeds 160, only the first 80 and last 80 elements are given.
4.1.2.2 Subset Properties
This section is generated during Fetch instructions, just before each subset is fetched. For each coordinate dimension of the variable to be fetched, the following information is given:

- **Start index**: Index of first element of subset
- **Input count**: Number of elements to be extracted
- **Stride**: Spacing between elements to be extracted
- **Straddle count**: Number of elements to skip at start of next year, if any
- **Start coordinate**: Coordinate value at first element of subset
- **End coordinate**: Coordinate value at last element of subset

4.1.2.3 Output Data Properties
This section is written to the log during Fetch instructions, just after each subset is fetched. It gives the name of the output file, the data type and name of the input variable, and the size and shape (that is, the total length and the length along each dimension) of the fetched subset. The resultant value which represents missing information in the output file, after scaling and offsetting the input value, is also given.
5 REFERENCES

