mseedcut(1)

Name

mseedcut - cut the miniSEED input into pieces

Synopsis

```
mseedcut  [-v|--verbose] [--include-pattern=PATTERN]...
        [--select-station=STATION]... [--select-channel=CHANNEL]...
        [--output-dir=DIRECTORY [--force-overwrite]
        [--force-concat] [--file-length=DURATION]]
        [--trace-start=TIMEMOMENT] [--trace-stop=TIMEMOMENT]
        [--trace-length=DURATION] [--trace-offset=SHIFT]
        [--events=EVENTFILE]
        [file | directory]...
```

```
mseedcut  [-h|--help] [--version] [--sysinfo]
```

Description

The **mseedcut** utility is used to cut out defined time windows from the miniSEED input as well as to split the input data into files of same length.

The program will either read from standard input or from files given as argument at the command line. If one or more directories are given at the command line as input, **mseedcut** searches recursively for files inside those directories. The search can be restricted to contain only files with a name also matched by a pattern given via one or more `--include-pattern` options.

To extract a specific time window from the miniSEED data stream the command line options `--trace-start`, `--trace-length`, `--trace-stop` and `--trace-offset` can be used. The time window is applied to all miniSEED input streams read. This makes it possible to cut out the same time window from many instruments and/or multiple channels in one program run.

The obtained cutouts are written to standard output (i.e. console) for further processing by other tools. If the `--output-dir` option is used the miniSEED output is redirected to one or more files in the respective directory. In addition, the miniSEED file output can automatically be written into files of equal length. Use the `--file-length` command line options to specify how long each miniSEED file should be.

Options

The program pretty much follows expected Unix command line syntax. Some command line options have two variants, one long and an additional short one (for convenience). These are shown below, separated by commas. However, most options only have a long variant. The '=' for options that take a parameter is required and can not be replaced by a whitespace.

-h, --help

Print a brief summary of all available command line options and exit.

--version

Print the **mseedcut** release information and exit.

--sysinfo

Provide some basic system information and exit.

-v, --verbose

This option increases the amount of information given to the user during program execution. By default, (i.e. without this option) **mseedcut** only reports warnings and errors. (See the diagnostics section below.)

--include-pattern=*PATTERN*

Only process files whose filename matches the given *PATTERN*. Files with a name not matching the search *PATTERN* will be ignored. This option is quite useful to speed up recursive searches through large subdirectory trees and can be used more than once in the same command line.

You can use the two wild card characters (* and ?) when specifying a *PATTERN* (e.g. `*.pri?`). Or alternatively, you can also use a predefined filter called `GIPP` that can be used to exclude all files not following the usual GIPP naming convention for miniSEED files recorded by Earth Data loggers (e.g. message logging or status files, see examples section below).

The search *PATTERN* is only applied to the filename part and not to the full pathname of a file.

--select-station=*STATION*

Only process miniSEED records with a matching *STATION* identifier. Records that do not match the *STATION* expression are ignored/skipped.

This command line option understands the two wild card character * and ? and can be used more than once in a command line.

--select-channel=*CHANNEL*

Only process miniSEED records with a matching *CHANNEL* identifier. Records that do not match the *CHANNEL* expression are ignored/skipped.

This command line option understands the two wild card character * and ? and can be used more than once in a command line.

--output-dir=*DIRECTORY*

Save the resulting miniSEED files containing the requested time series data to this *DIRECTORY*. The directory must already exist and be writable! Already existing files in that directory will not be overwritten unless the option `--force-overwrite` is used as well.

--force-overwrite

If this option is used, already existing files in the output directory will be overwritten without mercy!

The default behavior however is **not** to overwrite already existing files. Instead, a new file is created with an additional number in between filename and extension.

--force-concat

Concatenate the miniSEED output creating as few new files as possible. This means that a new output file is only started when there is a (data) discontinuity in the miniSEED input. Without discontinuity the converted data is simply appended to the currently used output file.

By default, however, a separate new output file is created for every single miniSEED input file.

--file-length=*DURATION*

Split the miniSEED output data into several files with a length of at most the given *DURATION*. Common values are `MINUTE`, `HOUR`, `DAY` or `WEEK`, which are also understood as input value (see examples section below). The **mseedcut** utility will start new output files at the full minute, hour, day or week respectively.

Alternatively, the *DURATION* can also be given directly as plain number (in seconds).

The following four command line options are all used to specify a time window for extracting data from the miniSEED input. It is considered an error to use `--trace-start`, `--trace-stop` and `--trace-length` together at the same time. At most two of the three options may be used for the same command line. Also, the option `--trace-length` cannot be used alone. It needs `--trace-start` or `--trace-stop` as anchor.

--trace-start=*TIMEMOMENT*

Begin extracting time series data at this moment in time. The format for the *TIMEMOMENT* string is *YYYY-MM-DD*T*HH:MM:SS.ssssss* where *YYYY-MM-DD* represents the date and *HH:MM:SS.ssssss* the time (fractions of seconds will be rounded to microsecond accuracy). The letter T in between date and time is used to distinguish between date and time part and must be given as well. Example: To begin reading samples at 1pm on March 27th, 2007 use the *TIMEMOMENT* string `--trace-start=2007-03-27T13:00:00`.

--trace-stop=*TIMEMOMENT*

Stop extracting time series data at this moment in time. The format for the *TIMEMOMENT* string is the same as with the `--trace-start` option.

--trace-length=*DURATION*

Stop extracting samples after this time span. The *DURATION* format is *SS.ssssss* and is given in seconds. Again, fractions of seconds are rounded to microsecond accuracy. Example: To extract 10 minutes of data use `--trace-length=600`.

A trace length of 5 minutes is used as default setting if no trace length option is given and a singular `--trace-start` or `--trace-stop` option is found.

--trace-offset=*SHIFT*

Use this option to shift the time window defined by the three command line options above. This option exists purely for convenience reasons as it would be easy to obtain the same effect by adding *SHIFT* seconds to the trace start and stop time manually. In other words, using `--trace-offset` just spares you from doing the math yourself in case you have a list of event times but would like to extract a few seconds of data before the event as well.

The format of the trace offset value is *SS.ssssss*, and it is given in seconds. Negative number shift the window towards earlier times, positive number "delay" the window. The total length of the time window is not affected by this option.

--events=*EVENTFILE*

In addition to the options described above it is also possible to use an event file to define time windows. Using an event file makes it possible to cut out more than one time window per program run. Each line in the event file must begin with the start time of the time window that should be read. Optionally, the length and offset of the time window may follow in the second and third column.

Event files contain up to three columns separated by spaces or tabulators. The three columns are:

Column #1

Start time of the time window. Analog to the `--trace-start` command line option. This column is mandatory.

Column #2

Length of the time window. Analog to the `--trace-length` command line option.

Column #3

An additional shift/offset applied to the time window. Analog to the `--trace-offset` command line option.

Empty lines in the file are ignored. Everything following a '#' character (up to the end of the line) is considered to be a comment and is skipped as well. Columns are counted from the beginning of the line. This means you cannot define a trace offset (column #3) without having a trace length (column #2) in the line first!

Environment

The following environment variables can optionally be used to influence the behavior of the GIPPtool utilities.

GIPPTOOLS_HOME

This environment variable is used to find the location of the GIPPtools installation directory. In particular, the Java class files that make up the GIPPtools are expected to be in the `java`

subdirectory of **GIPPTOOLS_HOME**.

GIPPTOOLS_JAVA

The utilities of the GIPPtools are written in the programming language Java and consequently need a Java Runtime Environment (JRE) to execute. Use this variable to specify the location of the JRE which should be used.

GIPPTOOLS_OPTS

You can use this environment variable for additional fine-tuning of the Java runtime environment. This is typically used to set the Java heap size available to GIPPtool programs.

It is usually not necessary to define any of those variables as suitable values should be selected automatically. However, if the automatic detection build into the start script fails, or you need to choose between different GIPPtool or Java runtime releases installed on your computer, these environment variables might become quite helpful to troubleshoot the situation.

Diagnostics

Occasionally, the **mseedcut** utility will produce user feedback. In general, user messages are classified as *INFO*, *WARNING* or *ERROR*. The *INFO* messages are only displayed when the `--verbose` command line option is used. They usually report about the progress of the program run, give statistical information or write a final summary.

More important are *WARNING* messages. In general, they warn about (possible) issues that may influence the outcome. Although the program will continue with execution, you certainly should check the results carefully. You might not have gotten what you (thought you) asked for. Finally, *ERROR* messages inform about problems that can not be resolved automatically. Program execution usually stops and the user must fix the cause of the error first.

Exit codes

Use the following program exit codes when calling **mseedcut** from scripts or other programs to see if **mseedcut** finished successfully. Any non-zero code indicates an *ERROR*!

0

Success.

64

Command line syntax or usage error.

65

Data format error. (The input was not a valid miniSEED file.)

66

An input file did not exist or was not readable.

74

I/O error.

99

Other, unspecified errors.

Examples

1. To extract a thirty-second long time series from an EDL miniSEED file beginning at 10:35pm on December 26th, 2007 use:

   ```
   mseedcut --trace-start=2007-12-26T22:35:00 --trace-length=30
   e3395071226233000.pri0 > ./result.mseed
   ```

2. To extract the same thirty-second long time window of the vertical component recorded by many stations run the following command line:

   ```
   mseedcut --trace-start=2007-12-26T22:35:00 --trace-length=30 --include
   -pattern=*.pri0 --output-dir=./event --force-concat  ./input
   ```

   This will search recursively through the subdirectory `./input` for miniSEED input files ending with the filename extension `.pri0` (the first primary recording channel). It will then return the requested time window, creating new, 30 seconds long miniSEED files in the `./event` subdirectory.

3. To simplify file handling it is often useful to create "day files", each containing the miniSEED data of one recorder channel and one day in a single file:

   ```
   mseedcut --file-length=86400 --output-dir=./daily  ./input
   ```

   Alternatively, you can also use one of the predefined keywords (*MINUTE*, *HOUR*, *DAY* or *WEEK*) instead:

   ```
   mseedcut --file-length=DAY --output-dir=./daily ./input
   ```

Files

$GIPPTOOLS_HOME/bin/mseedcut

The **mseedcut** "program". Usually just a copy of or symbolic link pointing to the standard GIPPtools start script.

$GIPPTOOLS_HOME/bin/gipptools

The GIPPtools start script. Almost all utilities of the GIPPtools package are started from this shell script.

See also

gipptools(1), **cube2ascii**(1), **cube2mseed**(1), **cube2segy**(1), **cubeevent**(1), **cubeinfo**(1), **mseed2ascii**(1), **mseed2mseed**(1), **mseed2pdas**(1), **mseed2segy**(1), **mseedinfo**(1), **mseedrecover**(1), **mseedrename**(1)

Bugs and caveats

- Extract data from *ASCII encoded* miniSEED records is not supported. There is no standard how the text contained in those records should be structured, which makes it difficult to read sample values. Also, *ASCII encoded* records often contain logging information, recorder configurations or other metadata instead of data samples.
  This makes it impossible to handle such files without custom software.