

cubeevent(1)

Name

cubeevent - list events captured by the Cube "event recorder" hardware

Synopsis

```
cubeevent [-v|--verbose] [--include-pattern=PATTERN]...  
          [--wnro-correction=AUTO|OFF|FORCE] [--format=FORMAT]  
          [--output-dir=DIRECTORY [--force-overwrite]]  
          file | directory...
```

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

Description

An *event recorder* is a specialized Cube data logger variant designed to record the precise time of discrete (seismic) events. Some typical sources for these events are sledgehammer, drop weight or explosives. The **cubeevent** program is used to readout information from *event recorder* files.

Calling **cubeevent** with one or more *event recorder files* as argument will by default return a short list of all recorded and detected events in those files. If a *directory* is given at the command line, **cubeevent** will search recursively for input files inside the directory. The report is by default written to standard output (i.e. console) or saved in an output directory (use option **--output-dir**). Different report variants (EVENTS, BATTERY, ...) are available and can be selected using the **--format** option.



The files written by *event recorder* hardware are essentially just a specialized variant of the "normal" Cube data logger file format. It is therefore feasible to use other GIPPtools such as e.g. **cubeinfo** or **cube2ascii** to access the contained time series data as well. However, only the **cubeevent** utility provides complete access to the extra information (e.g. automatically detected events or battery voltage) also contained in *event recorder* files.

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 cannot be replaced by a whitespace.

-h, --help

Summarize the most important command line options and exit.

--version

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

--sysinfo

Report system settings relevant for **cubeevent** 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) **cubeevent** 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 useful to speed up recursive searches through large subdirectory trees and can be used more than once in the same command line. (In this case files matching any of the provided patterns are included.)

You can use two wild card characters (*** and *?*) when specifying an include *PATTERN*. For example using *'*.123'* selects all files ending with the extension *.123*. Remember to enclose patterns with single quotes when using wildcards or spaces in the context of an *--include-pattern* command line option.

There is also a predefined GIPP filter. Using *--include-pattern=GIPP* automatically selects all data files following the default naming convention for files recorded by loggers borrowed from the Geophysical Instrument Pool Potsdam (GIPP).



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

--wnro-correction=AUTO|OFF|FORCE

Data loggers receive the recording time from the Global Positioning System (*GPS*). Unlike the commonly used Gregorian calendar, *GPS* (internally) expresses date/time information using two integer numbers. The first number counts the weeks since start of the *GPS* system on January 6th, 1980. The second number gives the seconds relative to the beginning of the week.

Satellites transmit the week number as a 10-bit long integer, which will become zero again every 1024 weeks. This integer overflow is called week number rollover (*WNRO*) and is a common issue with all GPS receivers.

Usually, the **cubeevent** utility will detect *WNRO* and just correct the date automatically by adding 1024 weeks to the "wrong" recording time. However, in cases where the automatic detection and correction fails, the handling of *WNRO* can be manually controlled using this command line option.

AUTO

Automatically detect and correct *WNRO* problems. This is the default mode.

OFF

Disable any (automatic) *WNRO* corrections for the input data. Time information will be used as recorded by the GPS receiver.

FORCE

Force apply *WNRO* corrections to the input data.

Please note that the `--wnro-correction` option only affects the handling of *WNRO* caused errors. General quality control of the timing accuracy can independently be adjusted using the `--timing-control` command line option.



Unless there is a good reason for it, the "*WNRO* correction" should be left in the default automatic mode. Use *OFF* or *FORCE* arguments to overwrite in case the automatic *WNRO* detection fails.

--output-dir=DIRECTORY

Save the resulting reports 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 before the file extension.

--format=FORMAT

Select one of the following predefined output formats:

EVENTS

List all recorded/triggered events. The output consists of a sequential event number, the event time and information about the age of the GPS fix that was used to determine the event time. Example:

```
# -----  
# recording unit: c0000      file name: 03301038.000  
# -----  
Event #1  2017-03-30T10:39:00.795750  (GPS ok)  
Event #2  2017-03-30T10:41:00.000031  (GPS 14s old)  
Event #3  2017-03-30T10:43:00.000313  (GPS ok)
```

Events are simply numbered in the order they were read from the file input. The information about the age of the GPS fix allows a rough assessment of the GPS reception during the

recording.



The **cubeevent** utility does not provide more detailed GPS information because this is already available via the **cubeinfo** program. Simply use the **--format=GPS** command line option of the **cubeinfo** utility.

If no **--format** command line option is used, the program will default to the *EVENTS* output format!

ALL

This mode will output *ALL* samples recorded by the *event recorder*. The output will consist of the recording time of the sample, the two primary recording channels (usually the electric signal from the cable used to trigger the explosion and a seismic signal from a geophone located nearby the source) as well as the two auxiliary channels tracking the state of the recording button (1 - pressed, 0 - unpressed) and marking the first sample after the detected event. Example:

```
# -----
# recording unit: c0000      file name: 03301038.000
# -----
2017-03-30T10:39:00.793000    -195  174  0  0
2017-03-30T10:39:00.794000    -124  -66  0  0
2017-03-30T10:39:00.795000     -88  476  1  0
2017-03-30T10:39:00.796000  -25122   97  1  0
2017-03-30T10:39:00.797000  -17719   80  1  1
2017-03-30T10:39:00.798000  -29410  421  1  0
2017-03-30T10:39:00.799000  -27118   41  1  0
2017-03-30T10:39:00.800000  -26366  121  0  0
2017-03-30T10:39:00.801000  -25775  313  0  0
```

REC

The output format is identical to the *ALL* format described above. However, only samples recorded while "recording" button was pressed (i.e. the value of the fourth column is 1) are written. This will reduce the returned information by the *ALL* output format to the "interesting parts".

BATTERY

Report the voltage of the internal battery over time. This is intended for diagnostic purposes only.

Environment

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

GIPPTOOLS_HOME

This environment variable can be used to set the GIPPTools installation directory.

In particular, the Java classes that make up the GIPTools are read from *JAR* files in the **java** subdirectory located inside **GIPPTOOLS_HOME**. The start scripts for the individual GIPTool utilities can be found inside the **bin** subdirectory.

GIPPTOOLS_JAVA

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

GIPPTOOLS_OPTS

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

GIPPTOOLS_LEAP

The GIPTools require up-to-date leap second information to correctly interpret Cube files. Usually, this information is read from the **leap-seconds.list** file located in the **config** subdirectory of the GIPTools installation directory (**GIPPTOOLS_HOME**). This environment variable can be used to provide a more up-to-date leap second list to GIPTool 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 GIPTool or Java runtime releases installed on your computer, these environment variables might become helpful to troubleshoot the situation.

Diagnostics

During execution, the **cubeevent** utility will produce user feedback. In general, user messages are classified as *INFO*, *WARNING* or *ERROR*.

INFO messages usually report about the progress of the program run, give statistical information or write a final summary. They are only displayed when the **--verbose** command line option is used.

More important are *WARNING* messages. In general, they warn about any issues that may influence the outcome in unexpected ways. 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 cannot 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 **cubeevent** from scripts or other programs to see if **cubeevent** finished successfully. Any non-zero code indicates an *ERROR*!

0

Success.

64

Command line syntax or usage error.

65

Input data was incorrectly formatted.

66

An input file did not exist or was not readable.

70

Error in internal program logic.

74

I/O error.

99

Other, unspecified errors.

Examples

1. To obtain a short list of all recorded and detected events contained in the *event recorder* file called `recording.cube`, you simply use the following:

```
cubeevent recording.cube
```

Files

`$GIPPTOOLS_HOME/bin/cubeevent`

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

`$GIPPTOOLS_HOME/bin/gipptools`

The standard start script used to run all GIPPTool utilities.

See also

`gipptools(1)`, `cube2ascii(1)`, `cube2mseed(1)`, `cube2segy(1)`, `cubeaux(1)`, `cubeevent(1)`, `cubeinfo(1)`, `cubeinspect(1)`, `mseed2ascii(1)`, `mseed2mseed(1)`, `mseed2pdas(1)`, `mseed2segy(1)`, `mseedcut(1)`, `mseedinfo(1)`, `mseedrecover(1)`, `mseedrename(1)`

Bugs and caveats

None so far.