cubeinfo(1)

Name

cubeinfo - summarize the content of a Cube recording

Synopsis

> cubeinfo [-v|--verbose] [--include-pattern=*PATTERN*]...
> [--output-dir=*DIRECTORY* [--force-overwrite]]
> [--format=*FORMAT*]
> [*file* | *directory*]...

> cubeinfo [-h|--help] [--version] [--sysinfo]

Description

Cubeinfo reads from a *file* (or standard input) and returns a short textual report of the content of the read Cube recording. If a *directory* is given instead, **cubeinfo** searches recursively for input files inside the directory.

The report is written to standard output (i.e. console) or saved in an output directory (use option `--output-dir`). Different report variants (*SUMMARY*, *DUMP*, ...) are predefined and can be selected using the `--format` option.

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 **cubeinfo** 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) **cubeinfo** 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. `*.123`). 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 files recorded by Cubes.

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

--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 in between filename and extension.

--format=*FORMAT*

Select one of the following predefined output formats:

INFO

Give a brief, minimal description of the content of the Cube recording. The report will contain the Cube instrument id and configuration, battery levels, as well as the approximate start and stop time of the recording. This is the default output format.

SUMMARY

A slightly more verbose description of the content also includes statistics about how many data blocks (header, trailer, samples, GPS, …) are contained in the Cube recording.

> This output format requires that the Cube file must be read completely, which may take a few seconds for extensive recordings.

GPS

List the content of all GPS blocks contained in the Cube recording. This output format is especially useful to obtain the actual position of a Cube while it was recording. You can also get a rough idea about the quality of the received GPS signal.

The returned information will depend on the build-in/connected clock hardware, which differs between Cube recorders. Usually the report contains at least the Cube block number and the time contained in the GPS block. Additional information is provided on an "as available" base and may include the number of leap seconds between GPS and UTC time (as reported by the GPS satellite and/or as officially announced by the International Earth Rotation and Reference Systems Service, IERS, the type of the GPS fix (2D, 3D, ...), the position of the Cube (latitude, longitude and maybe elevation), temperature (measured by the clock hardware), the number of satellites received, and/or the age of the GPS fix ("less than 10s old").

> The time information provided by the GPS output format is not identical to the recording time of any sample! Additional time corrections (depending amongst others on Cube hardware and configured sample rate) must be applied first to obtain the correct (precise) recording time.

DEBUG

Returns a textual representation of every data block contained in the Cube file. This rather voluminous report includes everything there is and should probably be used for debugging purpose only!

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.

GIPPTOOLS_LEAP

The GIPPtools require up-to-date leap second information to correctly interpret Cube files. Usually, this information is obtained from the `leap-seconds.list` file located in the config subdirectory of the GIPPtools installation directory. This environment variable can be used to provide a more up-to-date leap second list 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 **cubeinfo** 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 **cubeinfo** from scripts or other programs to see if **cubeinfo** 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 Cube recording.)

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 learn about the content of the single Cube file called `recording.cube`, you simply use one of the following:

```
cubeinfo  recording.cube
```

```
cubeinfo < recording.cube
```

> The first variant will usually be much faster as **cubeinfo** will jump directly from the beginning of the file, where it reads the "header block", to the end of the file where the "trailer block" required by the INFO report is located.
> The second variant does not have that option as it must always read the complete data stream piped in from console. There is, however, no significant difference when requesting a report format other than the default INFO format as the complete recording must be read for other predefined report formats.

2. To get information about how many data blocks are contained in the Cube file `02161251.034` use the *SUMMARY* output format:

   ```
   cubeinfo --format=SUMMARY  02161251.034
   ```

   This will not only return a table of the fields contained in the header and trailer block of the Cube file but also count every data block contained in the file and report those statistics too.

   > Using the *SUMMARY* output format is a convenient method to quickly check if the Cube file is "complete". If you suspect your Cube recording was cut-off early or might be otherwise damaged you can look for header, trailer and end-of-file block counts. A sound Cube file should contain exactly one of each!

3. To find out at which coordinates the Cube was placed when it recorded the file `02161251.034` you use *GPS* report format:

   ```
   cubeinfo --format=GPS  02161251.034
   ```

   The resulting output contains, besides other information, the coordinates (marked as *lat* and *lon*) of the Cube in degrees. Depending on the build-in GPS hardware additional information about elevation might or might not be available.

Files

$GIPPTOOLS_HOME/bin/cubeinfo

The **cubeinfo** "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), **mseed2ascii**(1), **mseed2mseed**(1), **mseed2pdas**(1), **mseed2segy**(1), **mseedcut**(1), **mseedinfo**(1), **mseedrecover**(1), **mseedrename**(1)

Bugs and caveats

- If a Cube file was cut-off early (i.e. it does not contain a trailer block at the end of the recording) no information about the end of the recording can be given! Although this should be obvious it always seems to surprise the user.