mseed2pdas(1)

Name

mseed2pdas - convert miniSEED data to PDAS format

Synopsis

> mseed2pdas [-v|--verbose] [--include-pattern=*PATTERN*]...
> [--select-station=*STATION*]... [--select-channel=*CHANNEL*]...
> [--output-dir=*DIRECTORY* [-force-overwrite] [--force-concat]]
> [*file* | *directory*]...

> mseed2pdas [-h|--help] [--version] [--sysinfo]

Description

Mseed2pdas reads from one or more *files* (or standard input) and converts each miniSEED record in the input to the file format used by the *Programmable Data Acquisition System* (*PDAS*) produced by Teledyne Geotech. If a *directory* is given instead, **mseed2pdas** searches recursively for input files inside the directory. 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.

The conversion result is written to standard output (i.e. console) or saved in an output directory (use option `--output-dir`).

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 **mseed2pdas** 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) **mseed2pdas** 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 PDAS files containing the converted miniSEED records to this *DIRECTORY*. The directory must already exist and be writable! Already existing PDAS 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 PDAS 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.

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 **mseed2pdas** 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 **mseed2pdas** from scripts or other programs to see if **mseed2pdas** 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 convert a single EDL miniSEED file to PDAS format you could use one of the following variations:

   ```
   mseed2pdas e3395071226233000.pri0 > ./p1395705.360
   ```

   ```
   mseed2pdas --output-dir=. e3395071226233000.pri0
   ```

   Both times the result will be a PDAS file called p1395705.360 in your current working directory.

2. You can also convert several miniSEED files in one run. The following will convert all EDL miniSEED files in directory eh-1234/040 to PDAS. The resulting files are saved to the ./pdas subdirectory.

   ```
   mseed2pdas --output-dir=./pdas eh-1234/040/*.pri?
   ```

   Alternatively, you can also work with the `--include-pattern` command line option. (Don't forget the single quotes around the pattern as it contains wild cards you do not want expanded by the shell!)

   ```
   mseed2pdas --output-dir=./pdas --include-pattern='*.pri?' eh-1234/040
   ```

   Or, instead of manually specifying a pattern, rely on the build in GIPP pattern:

   ```
   mseed2pdas --output-dir=./pdas --include-pattern=GIPP eh-1234/040
   ```

   The predefined GIPP pattern assumes that the miniSEED files follow the GIPP naming

convention for miniSEED files produced by EDL/EDR recorder. (Filenames start with a five character EDL/EDR unit id followed by a data and time stamp. The file extension indicates the recording channel. Example: `e3357080209143000.pri0`)

Whatever the command you use, in the end the ./pdas directory will contain one or more PDAS files. The number of files created depends on the miniSEED input. If the input is continuous (no gaps and no overlaps in the time series) only one PDAS file per recording channel will be created. However, if discontinuities in the input are detected, the **mseed2pdas** utility will start a new file.

3. You have a single miniSEED file contain several recording channels (e.g. *EHZ*, *EHN* and *EHE*) and are interested in the vertical component.

```
mseed2pdas --output-dir=./pdas --select-channel='??Z'  multiplexed.mseed
```

The difference between the `--include-pattern` option (used in example #2 above) and the two select command line options (`--select-station` and `--select-channel`) in this example is that `--include-pattern` is applied at the filesystem level. Only matching files are considered by **mseed2pdas**.

The select options, however, are applied while actually processing the input and can be used to pick individual records from the read miniSEED input.

This is most helpful when working with multiplexed miniSEED files containing several recording channels in one file.

Files

$GIPPTOOLS_HOME/bin/mseed2pdas

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

Bugs and caveats

None so far.