pysndfile Documentation

Release 1.4

Axel Roebel

Mar 10, 2022

Contents:

pysndfile is a python package providing *PySndfile*, a Cython wrapper class around libsndfile. PySndfile provides methods for reading and writing a large variety of soundfile formats on a variety of plattforms. PySndfile provides a rather complete access to the different sound file manipulation options that are available in libsndfile.

Due to the use of libsndfile nearly all sound file formats, (besides mp3 and derived formats) can be read and written with PySndfile.

The interface has been designed such that a rather large subset of the functionality of libsndfile can be used, notably the reading and writing of strings into soundfile formats that support these, and a number of sf_commands that allow to control the way libsndfile reads and writes the samples. One of the most important ones is the use of the clipping command.

CHAPTER 1

Transparent soundfile io with libsndfile

PySndfile has been developed in the analysis synthesis team at IRCAM mainly for research on sound analysis and sound transformation. In this context it is essential that the reading and writing of soundfile data is transparent.

The use of the clipping mode of libsndfile is important here because reading and writing sound data should not change the audio samples. By default, when clipping is off, libsndfile uses slightly different scaling factors when reading pcm format into float samples, or when writing float samples into pcm format. Therefore whenever a complete read/write cycle is applied to a sound file then the audio samples may be modified even when no processing is applied.

More precisely this will happen if

- the sound files contains pcm format,
- *and* the data is read into float or double,
- *and* the audio data comes close to the maximum range such that the difference in scaling leads to modification.

To avoid this problem PySndfile sets clipping by default to on. If you don't like this you can set it to off individually using the PySndfile method set_auto_clipping(False).

CHAPTER 2

Implementation

The implementation is based on a slightly modified version of the header sndfile.hh that is distributed with libsndfile. The only modification is the addition of a methode querying the seekable state of the open Sndfile.

CHAPTER 3

Installation

3.1 via Anaconda channel roebel

Precompiled packages are available for Anaconda python3 under Linux (x86_64) and Mac OS X (> 10.9). For these systems you can install pysndfile simply by means of

```
conda install -c roebel pysndfile
```

Unfortunately, I don't have a windows machine and therefore I cannot provide any packages for Windows.

3.2 compile with conda build recipe

You can use the conda recipe here. This build recipe wil automatically download and compile libsndfile building pysndfile.

3.3 via pypi

```
pip install pysndfile
```

should install pysndfile and python dependencies. Note, that pip cannot install libsndfile for you as it is not provided via pypi. To install libsndfile you should be able to use the software manager of your system. This will however only work if your software manager installs libsndfile such that the compiler used by the setup.py script will find it.

3.4 compile from sources

Note that for ompiling from sources you need to install requirements listed in requirements.txt file before starting the compilation. Moreover you need to install libsndfile as described in the previous section.

If the libsndfile (header and library) is not installed in the default compiler search path you have to specify the library and include directories to be added to this search paths. For this you can use either the command line options –sndfile-libdir and –sndfile-incdir that are available for the build_ext command or specify these two parameters in the setup.cfg file.

3.4.1 Windows

An experimental support for using pysndfile under windows has been added since version 1.3.4. For further comments see here as well as build scripts provided by sveinse. Note, that I do not have any windows machine and cannot provide any help in making this work.

CHAPTER 4

Documentation

Please see the developer documentation here.

CHAPTER 5

pysndfile package content

5.1 pysndfile package

5.1.1 pysndfile.sndio module

sndio is a simple interface for reading and writing arbitrary sound files. The module contains 3 functions.

functions:

pysndfile.sndio.get_info():: retrieve information from a sound file.
pysndfile.sndio.get_markers():: retrieve markers from aiff/ or wav files.
pysndfile.sndio.read():: read data and meta data from sound file.
pysndfile.sndio.write():: create a sound file from a given numpy array.

pysndfile.sndio.**get_info**(*name*, *extended_info=False*)
retrieve information from a sound file

Parameters

- **name** (*str*) – sndfile name
- **extended_info** (*bool*) –

Returns 3 or 5 tuple with meta information read from the soundfile in case extended_info is False a 3-tuple comprising samplerate, encoding (str), major format is returned in case extended_info is True a 5-tuple comprising additionally the number of frames and the number of channels is returned.

pysndfile.sndio.**get_markers**(*name*)
retrieve markers from sound file

Parameters **name** (*str*) – sound file name

Returns list of marker tuples containing the marker time and marker label.

Return type List

Note: following the implementation of libsndfile marker labels will be empty strings for all but aiff files.

pysndfile.sndio.**read**(*name*, *end=None*, *start=0*, *dtype=<class 'numpy.float64'>*, *return_format=False*, *sf_strings=None*, *force_2d=False*)

read samples from arbitrary sound files into a numpy array. May return subsets of samples as specified by start and end arguments (Def all samples) normalizes samples to [-1,1] if the datatype is a floating point type

The returned array is 1D for mono sound files and 2D with the channels in the columns for higher number of channels. If force_2d is given mono sound files will be returned in an array with shape (num_frames, 1)

Parameters

Parameters

- **name** (*str*) – sound file name
- **end** (*Union[int, None]*) – end sample frame position (not included into the segment to be returned) default=None -> read all samples
- **start** (*int*) – first sample frame to read default=0
- **dtype** (*numpy.dtype*) – data type of the numpy array that will be returned.
- **return_format** (*bool*) – if set then the return tuple will contain an additional element containing the sound file major format
- **sf_strings** (*Union[None,dict]*) – if a dict is given the dict elements will be set to the strings that are available in the sound file.
- **force_2d** (*bool*) – forces the returned array to have 2 dimensions with

Returns 3 or 4 -tuple containing data (1d for mon sounds 2d for multi channel sounds, where channels are in the columns), samplerate (int) and encoding (str), in case return_format is True then the next element contains the major format of the sound file (can be used to recreate a sound file with an identical format).

Return type Union[Tuple(numpy.array, int, str),Tuple(numpy.array, int, str, str)]

pysndfile.sndio.**write**(*name*, *data*, *rate=44100*, *format='aiff'*, *enc='pcm16'*, *sf_strings=None*)

Write datavector to sndfile using samplerate, format and encoding as specified valid format strings are all the keys in the dict pysndfile.fileformat_name_to_id valid encodings are those that are supported by the selected format from the list of keys in pysndfile.encoding_name_to_id.

Parameters

- **name** (*str*) – sndfile name
- **data** (*numpy.array*) – array containing sound data. For mono files an 1d array can be given, for multi channel sound files sound frames are in the rows and data channels in the columns.
- **rate** (*int*) – sample rate default s to 44100
- **format** (*str*) – sndfile major format default=aiff
- **enc** (*str*) – sndfile encoding default=pcm16
- **sf_strings** (*Union[dict, None]*) – dictionary containing bytes in ascii encoding to be written as meta data into the sound file. dictionary keys are limited to the keys available in *stringtype_name_to_id*‘ sf_strings arguments are only supported when the file format supports it. This are currently only the [aiff, wav, wavex, caf] formats. Note that each format imposes a particular limit to the length of individual strings. These lengths are stored

in the dict *max_supported_string_length*. If any of your strings exceeds the limit given in that dict a RuntimeError will be produced

Returns number of sample frames written.

Return type int

5.1.2 PySndfile wrapper class and methods

Mappings from libsndfile enums to pysndfile strings

pysndfile.**stringtype_name_to_id**
dict mapping of pysndfile's stringtype nams to libsndfile's stringtype ids.

pysndfile.**stringtype_id_to_name**
dict mapping of libsndfile's stringtype ids to pysndfile's stringtype names.

pysndfile.**commands_name_to_id**
dict mapping of pysndfile's commandtype names to libsndfile's commandtype ids.

pysndfile.**commands_id_to_name**
dict mapping of libsndfile's commandtype ids to pysndfile's commandtype names.

pysndfile.**endian_name_to_id**
dict mapping of pysndfile's endian names to libsndfile's endian ids.

pysndfile.**endian_id_to_name**
dict mapping of libsndfile's endian ids to pysndfile's endian names.

pysndfile.**fileformat_name_to_id**
dict mapping of pysndfile's fileformat names to libsndfile's major fileformat ids.

pysndfile.**fileformat_id_to_name**
dict mapping of libsndfile's major fileformat ids to pysndfile's major fileformat names.

Support functions

pysndfile.**construct_format**(*major*, *encoding*)
construct a format specification for libsndfile from major format string and encoding string

pysndfile.**get_pysndfile_version**()
return tuple describing the version of pysndfile

pysndfile.**get_sndfile_version**()
return a tuple of ints representing the version of libsndfile that is used

pysndfile.**get_sndfile_formats**()
Return lists of available file formats supported by libsndfile and pysndfile.

Returns list of strings representing all major sound formats that can be handled by the libsndfile library and the pysndfile interface.

pysndfile.**get_sndfile_encodings**(*major*)
Return lists of available encoding for the given sndfile format.

Parameters

Parameters **major** – (str) sndfile format for that the list of available encodings should be returned. format should be specified as a string, using one of the strings returned by *get_sndfile_formats()*

pysndfile.**get_sf_log**()
retrieve internal log from libsndfile, notably useful in case of errors.

Returns string representing the internal error log managed by libsndfile

PySndfile class

class pysndfile.**PySndfile**
Bases: object

PySndfile is a python class for reading/writing audio files.

PySndfile is proxy for the SndfileHandle class in sndfile.hh Once an instance is created, it can be used to read and/or write data from/to numpy arrays, query the audio file meta-data, etc...

Parameters

- **filename** – <string or int> name of the file to open (string), or file descriptor (integer)
- **mode** – <string> 'r' for read, 'w' for write, or 'rw' for read and write.
- **format** – <int> Required when opening a new file for writing, or to read raw audio files (without header). See function *construct_format()*.
- **channels** – <int> number of channels.
- **samplerate** – <int> sampling rate.

Returns valid PySndfile instance. An IOError exception is thrown if any error is encountered in libsndfile. A ValueError exception is raised if the arguments are invalid.

Notes

- the files will be opened with auto clipping set to True see the member set_autoclipping for more information.
- the soundfile will be closed when the class is destroyed
- format, channels and samplerate need to be given only in the write modes and for raw files.

channels(*self*)

Returns <int> number of channels of sndfile

close(*self*)
Closes file and deallocates internal structures

command(*self*, *command*, *arg=0*)

interface for passing commands via sf_command to underlying soundfile using sf_command(this_sndfile, command_id, NULL, arg)

param command <string or int> libsndfile command macro to be used. They can be specified either as string using the command macros name, or the command id.

Supported commands are:

SFC_SET_NORM_FLOAT
SFC_SET_NORM_DOUBLE

SFC_GET_NORM_FLOAT
SFC_GET_NORM_DOUBLE
SFC_SET_SCALE_FLOAT_INT_READ
SFC_SET_SCALE_INT_FLOAT_WRITE
SFC_SET_ADD_PEAK_CHUNK
SFC_UPDATE_HEADER_NOW
SFC_SET_UPDATE_HEADER_AUTO
SFC_SET_CLIPPING (see *pysndfile.PySndfile.set_auto_clipping()*)
SFC_GET_CLIPPING (see *pysndfile.PySndfile.set_auto_clipping()*)
SFC_WAVEX_GET_AMBISONIC
SFC_WAVEX_SET_AMBISONIC
SFC_RAW_NEEDS_ENDSWAP

param arg <int> additional argument of the command

return <int> 1 for success or True, 0 for failure or False

encoding_str(*self*)

Returns string representation of encoding (e.g. pcm16)

see *pysndfile.get_sndfile_encodings()* for a list of available encoding strings that are supported by a given sndfile format

error(*self*)
report error numbers related to the current sound file

format(*self*)

Returns <int> raw format specification that was used to create the present PySndfile instance.

frames(*self*)

Returns <int> number for frames (number of samples per channel)

get_cue_count(*self*)
get number of cue markers.

get_cue_mrks(*self*)
get all cue markers.

Gets list of tuple of positions and related names of embedded markers for aiff and wav files, due to a limited support of cue names in libsndfile cue names are not retrieved for wav files.

get_name(*self*)

Returns <str> filename that was used to open the underlying sndfile

get_strings(*self*)
get all stringtypes from the sound file.

see `stringtype_name_to_id` for the list of strings that are supported by the libsndfile version you use.

major_format_str(*self*)

Returns short string representation of major format (e.g. aiff)

see *pysndfile.get_sndfile_formats()* for a complete lst of fileformats

read_frames(*self*, *sf_count_t nframes=-1*, *dtype=np.float64*, *force_2d=False*, *fill_value=None*, *min_read=0*)

Read the given number of frames and put the data into a numpy array of the requested dtype.

Parameters

- **nframes** (*int*) – number of frames to read (default = -1 -> read all).
- **dtype** (*numpy.dtype*) – data type of the returned array containing read data (see note).
- **force_2d** (*bool*) – always return 2D arrays even if file is mono
- **fill_value** (*any tye that can be assigned to an array containing dtype*) – value to use for filling frames in case nframes is larger than the file
- **min_read** – when fill_value is not None and EOFError will be thrown when the number of read sample frames is equal to or lower than this value

Returns np.array<dtype> with sound data

Notes

- One column per channel.

rewind(*self*, *mode='rw'*)

rewind read/write/read and write position given by mode to start of file

samplerate(*self*)

Returns <int> samplerate

seek(*self*, *sf_count_t offset*, *int whence=C_SEEK_SET*, *mode='rw'*)

Seek into audio file: similar to python seek function, taking only in account audio data.

Parameters

- **offset** – <int> the number of frames (eg two samples for stereo files) to move relatively to position set by whence.
- **whence** – <int> only 0 (beginning), 1 (current) and 2 (end of the file) are valid.
- **mode** – <string> If set to 'rw', both read and write pointers are updated. If 'r' is given, only read pointer is updated, if 'w', only the write one is (this may of course make sense only if you open the file in a certain mode).

Returns <int> the number of frames from the beginning of the file

Notes

- Offset relative to audio data: meta-data are ignored.
- if an invalid seek is given (beyond or before the file), an IOError is raised; note that this is different from the seek method of a File object.

seekable(*self*)

Returns <bool> true for soundfiles that support seeking

set_auto_clipping(*self*, *arg=True*)

enable auto clipping when reading/writing samples from/to sndfile.

auto clipping is enabled by default. auto clipping is required by libsndfile to properly handle scaling between sndfiles with pcm encoding and float representation of the samples in numpy. When auto clipping is set to on reading pcm data into a float vector and writing it back with libsndfile will reproduce the original samples. If auto clipping is off, samples will be changed slightly as soon as the amplitude is close to the sample range because libsndfile applies slightly different scaling factors during read and write.

Parameters **arg** – <bool> indicator of the desired clipping state

Returns <int> 1 for success, 0 for failure

set_string(*self*, *stringtype_name*, *string*)
set one of the stringtype to the string given as argument. If you try to write a stringtype that is not supported by the library a RuntimeError will be raised If you try to write a string with length exceeding the length that can be read by libsndfile version 1.0.28 a RuntimeError will be raised as well these limits are stored in the dict max_supported_string_length.

set_strings(*self*, *sf_strings_dict*)
set all strings provided as key value pairs in sf_strings_dict. If you try to write a stringtype that is not supported by the library a RuntimeError will be raised. If you try to write a string with length exceeding the length that can be read by libsndfile version 1.0.28 a RuntimeError will be raised as well these limits are stored in the dict max_supported_string_length.

strError(*self*)
report error strings related to the current sound file

writeSync(*self*)
call the operating system's function to force the writing of all file cache buffers to disk the file.

No effect if file is open as read

write_frames(*self*, *ndarray input*)
write 1 or 2 dimensional array into sndfile.

Parameters **input** – <numpy array> containing data to write.

Returns int representing the number of frames that have been written

Notes

- One column per channel.
- updates the write pointer.
- if the input type is float, and the file encoding is an integer type, you should make sure the input data are normalized normalized data (that is in the range [-1..1] - which will corresponds to the maximum range allowed by the integer bitwidth).

CHAPTER 6

Copyright

CHAPTER 7

Author

Axel Roebel

CHAPTER 8

Credits

- Erik de Castro Lopo: for libsndfile
- David Cournapeau: for a few ideas I gathered from scikits.audiolab.
- The Cython maintainers for the efficient means to write interface definitions in Cython.

CHAPTER 9

Changes

9.1 Version_1.4.4 (2022-03-11)

- Fix for win32: improved error handling for PyUnicode_AsWideCharString (thanks to Andrey Bienkowski)

9.2 Version_1.4.3 (2020-01-20)

- changed sndio functions to all use PySndfile as context manager. This fixes the problem that the sndfile remains open when an error occurs which may in turn lead to inconsistencies if the sndfile is tried to be rewritten in an exception handler.

9.3 Version_1.4.2 (2019-12-18)

- fixed PySndfile.read_frames method to properly handle reading frames in parts (previous fix was incomplete)

9.4 Version_1.4.1 (2019-12-18)

- extended supported commands to change compression level when writing flac and ogg files
- fixed PySndfile.read_frames and sndio.read method to properly handle reading frames from the middle of a file

9.5 Version_1.4.0 (2019-12-17)

- Extended PySndfile class:
  - support use as context manager

  - added support for wve, ogg, MPC2000 sampler and RF64 wav files
  - added support for forcing to return 2D arrays even for mono files
  - added method to close the file and release all resources.
  - support reading more frames than present in the file using the fill_value for all values positioned after the end of the file

9.6 Version_1.3.8 (2019-10-22)

- (no changes in functionality)
- added documentation to distributed files
- added missing licence file to distribution
- thanks @toddrme2178 for patches.

9.7 Version_1.3.7 (2019-08-01)

- removed cython (a build requirement) from requirements.txt
- avoid cython warning and fix language_level in the .pyx source code
- add and support pre-release tags in the version number
- use hashlib to calculate the README checksum.
- fixed support for use with python 2.7 that was broken since 1.3.4

9.8 Version_1.3.6 (2019-07-27)

- fixed potential but undesired build dependency of pandoc
- added link to explanation for using pysndfile under windows
- fixed pandoc problem that does produce non ASCII chars in rst output.

9.9 Version_1.3.5 (2019-07-27)

- fixed two copy paste bug introduced in 1.3.4 1.3.4 did in fact not work at all :-(
- added a check target to the makefile that performs a complete built/install/test cycle to avoid problems as in 1.3.4

9.10 Version_1.3.4 (2019-07-23)

- added support for automatic installation of requirements
- remove precompiled cython source file and rely on pip requirements to provide cython so that cython compilation will always be possible.

- added experimental support for installation on win32 (thanks to Svein Seldal for the contributions).
- use expanduser for replacing ~ in filenames
- adapted cython source code to avoid all compiler warnings due to deprecated numpy api
- removed use of ez_setup.py that is no longer required.

9.11 Version_1.3.3 (2019-06-01)

- fixed missing command C_SFC_SET_SCALE_INT_FLOAT_WRITE (thanks to Svein Seldal for the bug report and fix)
- better documentation of sf_string-io in sndio.read and sndio.write
- limit size of strings to be written such that the written file can always be read back with libsndfile 1.0.28 (which imposes different constraints for different formats)
- better error handling when number of channels exceeds channel limit imposed by libsndfile.
- sndio module now exposes the dicts: fileformat_name_to_id and fileformat_id_to_name
- extended sndio.read with force_2d argument that can be used to force the returned data array to always have 2 dimensions even for mono files.

9.12 Version_1.3.2 (2018-07-04)

- fixed documentation of sndio module.

9.13 Version_1.3.1 (2018-07-04)

- Extended sndio by means of adding a enw function that allows retrieving embed markers from sound files. Names marker labels will be retrieved only for aiff files.
- removed print out in pysndfile.get_cue_mrks(self) function.
- fixed version number in documentation.

9.14 Version_1.3.0 (2018-07-04)

- Added support for retrieving cue points from aiff and wav files.

9.15 Version_1.2.2 (2018-06-11)

- fixed c++-include file that was inadvertently scrambled.

9.16 Version_1.2.1 (2018-06-11)

- fixed formatting error in long description and README.
- setup.py to explicitly select formatting of the long description.

9.17 Version_1.2.0 (2018-06-11)

- support reading and writing sound file strings in sndio module
- Improved documentation of module constant mappings and PySndfile methods.
- Added a new method supporting to write all strings in a dictionary to the sound file.

9.18 Version_1.1.1 (2018-06-10)

this update is purely administrative, no code changes

- moved project to IRCAM GitLab
- moved doc to ReadTheDoc
- fixed documentation.

9.19 Version_1.1.0 (2018-02-13)

- support returning extended sndfile info covering number of frames and number of channels from function sndio.get_info.

9.20 Version_1.0.0 (2017-07-26)

- Updated version number to 1.0.0:
- pysndfile has now been used for quiet a while under python 3 and most problems seem to be fixed.
- changed setup.py to avoid uploading outdated LONG_DESC file.

9.21 Version_0.2.15 (2017-07-26)

- fixed get_sndfile_version function and tests script: adapted char handling to be compatible with python 3.

9.22 Version 0.2.14 (2017-07-26)

- fixed filename display in warning messages due to invalid pointer: replaced char* by std::string

9.23 Version 0.2.13 (2017-06-03)

- fixed using "~" for representing $HOME in filenames:
- _pysndfile.pyx: replaced using cython getenv by os.environ to avoid type incompatibilities in python3

9.24 Version 0.2.12 (2017-05-11)

- fixed problem in sndio.read: Optionally return full information required to store the file using the corresponding write function
- _pysndfile.pyx: add constants SF_FORMAT_TYPEMASK and SF_FORMAT_SUBMASK, SF_FORMAT_ENDMASK to python interface Added new function for getting internal sf log in case of errors. Improved consistency of variable definitions by means of retrieving them automatically from sndfile.h

9.25 Version 0.2.11 (2015-05-17)

- setup.py: fixed problem with compilers not providing the compiler attribute (MSVC) (Thanks to Felix Hanke for reporting the problem)
- _pysndfile.pyx: fixed problem when deriving from PySndfile using a modified list of **init** parameters in the derived class (Thanks to Sam Perry for the suggestion).

9.26 Version 0.2.10

- setup.py: rebuild LONG_DESC only if sdist method is used.

9.27 Version 0.2.9

- Added missing files to distribution.
- force current cythonized version to be distributed.

9.28 Version 0.2.4

- Compatibility with python 3 (thanks to Eduardo Moguillansky)
- bug fix: ensure that vectors returned by read_frames function own their data.

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index