# **sf\_command**

int sf\_command (SNDFILE \*sndfile, int cmd, void \*data, int datasize) ;

This function allows the caller to retrieve information from or change aspects of the library behaviour. Examples include retrieving a string containing the library version or changing the scaling applied to floating point sample data during read and write. Most of these operations are performed on a per-file basis.

The cmd parameter is a integer identifier which is defined in <sndfile.h>. All of the valid command identifiers have names begining with "SFC\_". Data is passed to and returned from the library by use of a void pointer. The library will not read or write more than datasize bytes from the void pointer. For some calls no data is required in which case data should be NULL and datasize may be used for some other purpose.

The available commands are as follows:

|  |  |
| --- | --- |
| [SFC\_GET\_LIB\_VERSION](#gjdgxs) | Retrieve the version of the library. |
| [SFC\_GET\_LOG\_INFO](#30j0zll) | Retrieve the internal per-file operation log. |
| [SFC\_CALC\_SIGNAL\_MAX](#1fob9te) | Retrieve the measured maximum signal value. |
| [SFC\_CALC\_NORM\_SIGNAL\_MAX](#3znysh7) | Retrieve the measured normalised maximum signal value. |
| [SFC\_CALC\_MAX\_ALL\_CHANNELS](#2et92p0) | Calculate peaks for all channels. |
| [SFC\_CALC\_NORM\_MAX\_ALL\_CHANNELS](#tyjcwt) | Calculate normalised peaks for all channels. |
| [SFC\_SET\_NORM\_FLOAT](#3dy6vkm) | Modify the normalisation behaviour of the floating point reading and writing functions. |
| [SFC\_SET\_NORM\_DOUBLE](#1t3h5sf) | Modify the normalisation behaviour of the double precision floating point reading and writing functions. |
| [SFC\_GET\_NORM\_FLOAT](#4d34og8) | Retrieve the current normalisation behaviour of the floating point reading and writing functions. |
| [SFC\_GET\_NORM\_DOUBLE](#2s8eyo1) | Retrieve the current normalisation behaviour of the double precision floating point reading and writing functions. |
| [SFC\_GET\_SIMPLE\_FORMAT\_COUNT](#17dp8vu) | Retrieve the number of simple formats supported by libsndfile. |
| [SFC\_GET\_SIMPLE\_FORMAT](#3rdcrjn) | Retrieve information about a simple format. |
| [SFC\_GET\_FORMAT\_INFO](#26in1rg) | Retrieve information about a major or subtype format. |
| [SFC\_GET\_FORMAT\_MAJOR\_COUNT](#lnxbz9) | Retrieve the number of major formats. |
| [SFC\_GET\_FORMAT\_MAJOR](#35nkun2) | Retrieve information about a major format type. |
| [SFC\_GET\_FORMAT\_SUBTYPE\_COUNT](#1ksv4uv) | Retrieve the number of subformats. |
| [SFC\_GET\_FORMAT\_SUBTYPE](#44sinio) | Retrieve information about a subformat. |
| [SFC\_SET\_ADD\_PEAK\_CHUNK](#2jxsxqh) | Switch the code for adding the PEAK chunk to WAV and AIFF files on or off. |
| [SFC\_UPDATE\_HEADER\_NOW](#z337ya) | Used when a file is open for write, this command will update the file header to reflect the data written so far. |
| [SFC\_SET\_UPDATE\_HEADER\_AUTO](#3j2qqm3) | Used when a file is open for write, this command will cause the file header to be updated after each write to the file. |
| [SFC\_FILE\_TRUNCATE](#1y810tw) | Truncate a file open for write or for read/write. |
| [SFC\_SET\_RAW\_START\_OFFSET](#4i7ojhp) | Change the data start offset for files opened up as SF\_FORMAT\_RAW. |

## 

**SFC\_GET\_LIB\_VERSION**

Retrieve the version of the library as a string.

Parameters:

sndfile : Not used  
 cmd : SFC\_GET\_LIB\_VERSION  
 data : A pointer to a char buffer  
 datasize : The size of the the buffer

Example:

char buffer [128] ;  
 sf\_command (NULL, SFC\_GET\_LIB\_VERSION, buffer, sizeof (buffer)) ;

Return value: This call will return the length of the retrieved version string. Notes: The string returned in the buffer passed to this function will not overflow the buffer and will always be null terminated .

**SFC\_GET\_LOG\_INFO**

Retrieve the log buffer generated when opening a file as a string. This log buffer can often contain a good reason for why libsndfile failed to open a particular file.

Parameters:

sndfile : A valid SNDFILE\* pointer  
 cmd : SFC\_GET\_LOG\_INFO  
 data : A pointer to a char buffer  
 datasize : The size of the the buffer

Example:

char buffer [2048] ;  
 sf\_command (sndfile, SFC\_GET\_LOG\_INFO, buffer, sizeof (buffer)) ;

Return value: This call will return the length of the retrieved version string. Notes: The string returned in the buffer passed to this function will not overflow the buffer and will always be null terminated .

**SFC\_CALC\_SIGNAL\_MAX**

Retrieve the measured maximum signal value. This involves reading through the whole file which can be slow on large files.

Parameters:

sndfile : A valid SNDFILE\* pointer  
 cmd : SFC\_CALC\_SIGNAL\_MAX  
 data : A pointer to a double  
 datasize : sizeof (double)

Example:

double max\_val ;  
 sf\_command (sndfile, SFC\_CALC\_SIGNAL\_MAX, &max\_val, sizeof (max\_val)) ;

Return value: Zero on success, non-zero otherwise.

**SFC\_CALC\_NORM\_SIGNAL\_MAX**

Retrieve the measured normailised maximum signal value. This involves reading through the whole file which can be slow on large files.

Parameters:

sndfile : A valid SNDFILE\* pointer  
 cmd : SFC\_CALC\_NORM\_SIGNAL\_MAX  
 data : A pointer to a double  
 datasize : sizeof (double)

Example:

double max\_val ;  
 sf\_command (sndfile, SFC\_CALC\_NORM\_SIGNAL\_MAX, &max\_val, sizeof (max\_val)) ;

Return value: Zero on success, non-zero otherwise.

**SFC\_CALC\_MAX\_ALL\_CHANNELS**

Calculate peaks for all channels. This involves reading through the whole file which can be slow on large files.

Parameters:

sndfile : A valid SNDFILE\* pointer  
 cmd : SFC\_CALC\_MAX\_ALL\_CHANNELS  
 data : A pointer to a double  
 datasize : sizeof (double) \* number\_of\_channels

Example:

double peaks [number\_of\_channels] ;  
 sf\_command (sndfile, SFC\_CALC\_MAX\_ALL\_CHANNELS, peaks, sizeof (peaks)) ;

Return value: Zero if peaks have been calculated successfully and non-zero otherwise.

**SFC\_CALC\_NORM\_MAX\_ALL\_CHANNELS**

Calculate normalised peaks for all channels. This involves reading through the whole file which can be slow on large files.

Parameters:

sndfile : A valid SNDFILE\* pointer  
 cmd : SFC\_CALC\_NORM\_MAX\_ALL\_CHANNELS  
 data : A pointer to a double  
 datasize : sizeof (double) \* number\_of\_channels

Example:

double peaks [number\_of\_channels] ;  
 sf\_command (sndfile, SFC\_CALC\_NORM\_MAX\_ALL\_CHANNELS, peaks, sizeof (peaks)) ;

Return value: Zero if peaks have been calculated successfully and non-zero otherwise.

**SFC\_SET\_NORM\_FLOAT**

This command only affects data read from or written to using the floating point functions:

size\_t [sf\_read\_float](http://docs.google.com/api.html#read) (SNDFILE \*sndfile, float \*ptr, size\_t items) ;  
 size\_t [sf\_readf\_float](http://docs.google.com/api.html#readf) (SNDFILE \*sndfile, float \*ptr, size\_t frames) ;  
  
 size\_t [sf\_write\_float](http://docs.google.com/api.html#write) (SNDFILE \*sndfile, float \*ptr, size\_t items) ;  
 size\_t [sf\_writef\_float](http://docs.google.com/api.html#writef) (SNDFILE \*sndfile, float \*ptr, size\_t frames) ;

Parameters:

sndfile : A valid SNDFILE\* pointer  
 cmd : SFC\_SET\_NORM\_FLOAT  
 data : NULL  
 datasize : SF\_TRUE or SF\_FALSE

For read operations setting normalisation to SF\_TRUE means that the data from all subsequent reads will be be normalised to the range [-1.0, 1.0].

For write operations, setting normalisation to SF\_TRUE means than all data supplied to the float write functions should be in the range [-1.0, 1.0] and will be scaled for the file format as necessary.

For both cases, setting normalisation to SF\_FALSE means that no scaling will take place.

Example:

sf\_command (sndfile, SFC\_SET\_NORM\_FLOAT, NULL, SF\_TRUE) ;  
  
 sf\_command (sndfile, SFC\_SET\_NORM\_FLOAT, NULL, SF\_FALSE) ;

Return value: Returns 1 on success or 0 for failure.

**SFC\_SET\_NORM\_DOUBLE**

This command only affects data read from or written to using the double precision floating point functions:

size\_t [sf\_read\_double](http://docs.google.com/api.html#read) (SNDFILE \*sndfile, double \*ptr, size\_t items) ;  
 size\_t [sf\_readf\_double](http://docs.google.com/api.html#readf) (SNDFILE \*sndfile, double \*ptr, size\_t frames) ;  
  
 size\_t [sf\_write\_double](http://docs.google.com/api.html#write) (SNDFILE \*sndfile, double \*ptr, size\_t items) ;  
 size\_t [sf\_writef\_double](http://docs.google.com/api.html#writef) (SNDFILE \*sndfile, double \*ptr, size\_t frames) ;

Parameters:

sndfile : A valid SNDFILE\* pointer  
 cmd : SFC\_SET\_NORM\_DOUBLE  
 data : NULL  
 datasize : SF\_TRUE or SF\_FALSE

For read operations setting normalisation to SF\_TRUE means that the data from all subsequent reads will be be normalised to the range [-1.0, 1.0].

For write operations, setting normalisation to SF\_TRUE means than all data supplied to the double write functions should be in the range [-1.0, 1.0] and will be scaled for the file format as necessary.

For both cases, setting normalisation to SF\_FALSE means that no scaling will take place.

Example:

sf\_command (sndfile, SFC\_SET\_NORM\_DOUBLE, NULL, SF\_TRUE) ;  
  
 sf\_command (sndfile, SFC\_SET\_NORM\_DOUBLE, NULL, SF\_FALSE) ;

Return value: Returns 1 on success or 0 for failure.

**SFC\_GET\_NORM\_FLOAT**

Retrieve the current float normalisation mode.

Parameters:

sndfile : A valid SNDFILE\* pointer  
 cmd : SFC\_GET\_NORM\_FLOAT  
 data : NULL  
 datasize : anything

Example:

normalisation = sf\_command (sndfile, SFC\_GET\_NORM\_FLOAT, NULL, 0) ;

Return value: Returns TRUE if normaisation is on and FALSE otherwise.

**SFC\_GET\_NORM\_DOUBLE**

Retrieve the current float normalisation mode.

Parameters:

sndfile : A valid SNDFILE\* pointer  
 cmd : SFC\_GET\_NORM\_DOUBLE  
 data : NULL  
 datasize : anything

Example:

normalisation = sf\_command (sndfile, SFC\_GET\_NORM\_DOUBLE, NULL, 0) ;

Return value: Returns TRUE if normalisation is on and FALSE otherwise.

**SFC\_GET\_SIMPLE\_FORMAT\_COUNT**

Retrieve the number of simple formats supported by libsndfile.

Parameters:

sndfile : Not used.  
 cmd : SFC\_GET\_SIMPLE\_FORMAT\_COUNT  
 data : a pointer to an int  
 datasize : sizeof (int)

Example:

int count ;  
 sf\_command (sndfile, SFC\_GET\_SIMPLE\_FORMAT\_COUNT, &count, sizeof (int)) ;

Return value: 0

**SFC\_GET\_SIMPLE\_FORMAT**

Retrieve information about a simple format.

Parameters:

sndfile : Not used.  
 cmd : SFC\_GET\_SIMPLE\_FORMAT  
 data : a pointer to an SF\_FORMAT\_INFO struct  
 datasize : sizeof (SF\_FORMAT\_INFO)

The SF\_FORMAT\_INFO struct is defined in <sndfile.h> as:

typedef struct  
 { int format ;  
 const char \*name ;  
 const char \*extension ;  
 } SF\_FORMAT\_INFO ;

When sf\_command() is called with SF\_GET\_SIMPLE\_FORMAT, the value of the format field should be the format number (ie 0 <= format <= count value obtained using SF\_GET\_SIMPLE\_FORMAT\_COUNT).

Example:

SF\_FORMAT\_INFO format\_info ;  
 int k, count ;  
  
 sf\_command (sndfile, SFC\_GET\_SIMPLE\_FORMAT\_COUNT, &count, sizeof (int)) ;  
  
 for (k = 0 ; k < count ; k++)  
 { format\_info.format = k ;  
 sf\_command (sndfile, SFC\_GET\_SIMPLE\_FORMAT, &format\_info, sizeof (format\_info)) ;  
 printf ("%08x %s %s\n", format\_info.format, format\_info.name, format\_info.extension) ;  
 } ;

Return value: 0 on success and non-zero otherwise. The value of the format field of the SF\_FORMAT\_INFO struct will be an value which can be placed in the format field of an SF\_INFO struct when a file is to be opened for write. The name field will contain a char\* pointer to the name of the string ie "WAV (Microsoft 16 bit PCM)". The extention field will contain the most commonly used file extension for that file type.

**SFC\_GET\_FORMAT\_INFO**

Retrieve information about a major or subtype format.

Parameters:

sndfile : Not used.  
 cmd : SFC\_GET\_FORMAT\_INFO  
 data : a pointer to an SF\_FORMAT\_INFO struct  
 datasize : sizeof (SF\_FORMAT\_INFO)

The SF\_FORMAT\_INFO struct is defined in <sndfile.h> as:

typedef struct  
 { int format ;  
 const char \*name ;  
 const char \*extension ;  
 } SF\_FORMAT\_INFO ;

When sf\_command() is called with SF\_GET\_FORMAT\_INFO, the format field is examined and if (format & SF\_FORMAT\_TYPEMASK) is a valid format then the struct is filled in with information about the given major type. If (format & SF\_FORMAT\_TYPEMASK) is FALSE and (format & SF\_FORMAT\_SUBMASK) is a valid subtype format then the struct is filled in with information about the given subtype.

Example:

SF\_FORMAT\_INFO format\_info ;  
  
 format\_info.format = SF\_FORMAT\_WAV ;  
 sf\_command (sndfile, SFC\_GET\_FORMAT\_INFO, &format\_info, sizeof (format\_info)) ;  
 printf ("%08x %s %s\n", format\_info.format, format\_info.name, format\_info.extension) ;  
  
 format\_info.format = SF\_FORMAT\_ULAW ;  
 sf\_command (sndfile, SFC\_GET\_FORMAT\_INFO, &format\_info, sizeof (format\_info)) ;  
 printf ("%08x %s\n", format\_info.format, format\_info.name) ;

Return value: 0 on success and non-zero otherwise.

**SFC\_GET\_FORMAT\_MAJOR\_COUNT**

Retrieve the number of major formats.

Parameters:

sndfile : Not used.  
 cmd : SFC\_GET\_FORMAT\_MAJOR\_COUNT  
 data : a pointer to an int  
 datasize : sizeof (int)

Example:

int count ;  
 sf\_command (sndfile, SFC\_GET\_FORMAT\_MAJOR\_COUNT, &count, sizeof (int)) ;

Return value: 0

**SFC\_GET\_FORMAT\_MAJOR**

Retrieve information about a major format type.

Parameters:

sndfile : Not used.  
 cmd : SFC\_GET\_FORMAT\_MAJOR  
 data : a pointer to an SF\_FORMAT\_INFO struct  
 datasize : sizeof (SF\_FORMAT\_INFO)

Example:

SF\_FORMAT\_INFO format\_info ;  
 int k, count ;  
  
 sf\_command (sndfile, SFC\_GET\_FORMAT\_MAJOR\_COUNT, &count, sizeof (int)) ;  
  
 for (k = 0 ; k < count ; k++)  
 { format\_info.format = k ;  
 sf\_command (sndfile, SFC\_GET\_FORMAT\_MAJOR, &format\_info, sizeof (format\_info)) ;  
 printf ("%08x %s %s\n", format\_info.format, format\_info.name, format\_info.extension) ;  
 } ;

For a more comprehensive example, see the program list\_formats.c in the examples/ directory of the libsndfile source code distribution.

Return value: 0 on success and non-zero otherwise. The value of the format field will one of the major format identifiers suc as SF\_FORMAT\_WAV SF\_FORMAT\_AIFF. The name field will contain a char\* pointer to the name of the string ie "WAV (Microsoft)". The extention field will contain the most commonly used file extension for that file type.

**SFC\_GET\_FORMAT\_SUBTYPE\_COUNT**

Retrieve the number of subformats.

Parameters:

sndfile : Not used.  
 cmd : SFC\_GET\_FORMAT\_SUBTYPE\_COUNT  
 data : a pointer to an int  
 datasize : sizeof (int)

Example:

int count ;  
 sf\_command (sndfile, SFC\_GET\_FORMAT\_SUBTYPE\_COUNT, &count, sizeof (int)) ;

Return value: 0

**SFC\_GET\_FORMAT\_SUBTYPE**

Retrieve information about a subformat.

Parameters:

sndfile : Not used.  
 cmd : SFC\_GET\_FORMAT\_SUBTYPE  
 data : a pointer to an SF\_FORMAT\_INFO struct  
 datasize : sizeof (SF\_FORMAT\_INFO)

Example:

SF\_FORMAT\_INFO format\_info ;  
 int k, count ;  
  
 sf\_command (sndfile, SFC\_GET\_FORMAT\_SUBTYPE\_COUNT, &count, sizeof (int)) ;  
  
 /\* Retrieve all the subtypes supported by the WAV format. \*/  
 for (k = 0 ; k < count ; k++)  
 { format\_info.format = k ;  
 sf\_command (sndfile, SFC\_GET\_FORMAT\_SUBTYPE, &format\_info, sizeof (format\_info)) ;  
 if (! sf\_format\_check (format.info | SF\_FORMAT\_WAV))  
 continue ;  
 printf ("%08x %s\n", format\_info.format, format\_info.name) ;  
 } ;

For a more comprehensive example, see the program list\_formats.c in the examples/ directory of the libsndfile source code distribution.

Return value: 0 on success and non-zero otherwise. The value of the format field will one of the major format identifiers such as SF\_FORMAT\_WAV SF\_FORMAT\_AIFF. The name field will contain a char\* pointer to the name of the string; for instance "WAV (Microsoft)" or "AIFF (Apple/SGI)". The extention field will be a NULL pointer.

**SFC\_SET\_ADD\_PEAK\_CHUNK**

By default, WAV and AIFF files which contain floating point data (subtype SF\_FORMAT\_FLOAT or SF\_FORMAT\_DOUBLE) have a PEAK chunk. By using this command, the addition of a PEAK chunk can be turned on or off.

Note : This call must be made before any data is written to the file.

Parameters:

sndfile : A valid SNDFILE\* pointer  
 cmd : SFC\_SET\_ADD\_PEAK\_CHUNK  
 data : Not used (should be NULL)  
 datasize : TRUE or FALSE.

Example:

/\* Turn on the PEAK chunk. \*/  
 sf\_command (sndfile, SFC\_SET\_ADD\_PEAK\_CHUNK, NULL, SF\_TRUE) ;  
  
 /\* Turn off the PEAK chunk. \*/  
 sf\_command (sndfile, SFC\_SET\_ADD\_PEAK\_CHUNK, NULL, SF\_FALSE) ;

Return value: Returns SF\_TRUE if the peak chunk will be written after this call. Returns SF\_FALSE if the peak chunk will not be written after this call.

**SFC\_UPDATE\_HEADER\_NOW**

The header of an audio file is normally written by libsndfile when the file is closed using **sf\_close()**.

There are however situations where large files are being generated and it would be nice to have valid data in the header before the file is complete. Using this command will update the file header to reflect the amount of data written to the file so far. Other programs opening the file for read (before any more data is written) will then read a valid sound file header.

Parameters:

sndfile : A valid SNDFILE\* pointer  
 cmd : SFC\_UPDATE\_HEADER\_NOW  
 data : Not used (should be NULL)  
 datasize : Not used.

Example:

/\* Update the header now. \*/  
 sf\_command (sndfile, SFC\_UPDATE\_HEADER\_NOW, NULL, 0) ;

Return value: 0

**SFC\_SET\_UPDATE\_HEADER\_AUTO**

Similar to SFC\_UPDATE\_HEADER\_NOW but updates the header at the end of every call to the **sf\_write\*** functions.

Parameters:

sndfile : A valid SNDFILE\* pointer  
 cmd : SFC\_UPDATE\_HEADER\_NOW  
 data : Not used (should be NULL)  
 datasize : SF\_TRUE or SF\_FALSE

Example:

/\* Turn on auto header update. \*/  
 sf\_command (sndfile, SFC\_SET\_UPDATE\_HEADER\_AUTO, NULL, SF\_TRUE) ;  
  
 /\* Turn off auto header update. \*/  
 sf\_command (sndfile, SFC\_SET\_UPDATE\_HEADER\_AUTO, NULL, SF\_FALSE) ;

Return value: TRUE if auto update header is now on; FALSE otherwise.

**SFC\_FILE\_TRUNCATE**

Truncate a file open for write or for read/write.

Parameters:

sndfile : A valid SNDFILE\* pointer  
 cmd : SFC\_FILE\_TRUNCATE  
 data : A pointer to an sf\_count\_t.  
 datasize : sizeof (sf\_count\_t)

Truncate the file to the number of frames specified by the sf\_count\_t pointed to by data. After this command, both the read and the write pointer will be at the new end of the file. This command will fail (returning non-zero) if the requested truncate position is beyond the end of the file.

Example:

/\* Truncate the file to a length of 20 frames. \*/  
 sf\_count\_t frames = 20 ;  
 sf\_command (sndfile, SFC\_FILE\_TRUNCATE, &frames, sizeof (frames)) ;

Return value: Zero on sucess, non-zero otherwise.

**SFC\_SET\_RAW\_START\_OFFSET**

Change the data start offset for files opened up as SF\_FORMAT\_RAW.

Parameters:

sndfile : A valid SNDFILE\* pointer  
 cmd : SFC\_SET\_RAW\_START\_OFFSET  
 data : A pointer to an sf\_count\_t.  
 datasize : sizeof (sf\_count\_t)

For a file opened as format SF\_FORMAT\_RAW, set the data offset to the value given by data.

Example:

/\* Reset the data offset to 5 bytes from the start of the file. \*/  
 sf\_count\_t offset = 5 ;  
 sf\_command (sndfile, SFC\_SET\_RAW\_START\_OFFSET, &offset, sizeof (offset)) ;

Return value: Zero on sucess, non-zero otherwise.

The libsndfile home page is here : <http://www.mega-nerd.com/libsndfile/>.

Version : 1.0.24