NAME

zip_source_function, **zip_source_function_create** - create data source from function

LIBRARY

libzip (-lzip)

SYNOPSIS

#include <zip.h>

*zip_source_t **
zip_source_function(*zip_t *archive, zip_source_callback fn, void *userdata*);

*zip_source_t **
zip_source_function_create(*zip_source_callback fn, void *userdata, zip_error_t *error*);

DESCRIPTION

The functions **zip_source_function**() and **zip_source_function_create**() create a zip source from the user-provided function *fn*, which must be of the following type:

typedef zip_int64_t (***zip_source_callback**)(*void *userdata, void *data, zip_uint64_t len, zip_source_cmd_t cmd*)

archive or *error* are used for reporting errors and can be NULL.

When called by the library, the first argument is the *userdata* argument supplied to the function. The next two arguments are a buffer *data* of size *len* when data is passed in or expected to be returned, or else NULL and 0. The last argument, *cmd*, specifies which action the function should perform.

Depending on the uses, there are three useful sets of commands to be supported by a **zip_source_callback**():

read source	Providing streamed data (for file data added to archives). Must support ZIP_SOURCE_OPEN, ZIP_SOURCE_READ, ZIP_SOURCE_CLOSE, ZIP_SOURCE_STAT, and ZIP_SOURCE_ERROR. If your source uses any allocated memory (including *userdata*) it should also implement ZIP_SOURCE_FREE to avoid memory leaks.
seekable read source	Same as previous, but from a source allowing reading from arbitrary offsets (also for read-only zip archive). Must additionally support

	ZIP_SOURCE_SEEK, ZIP_SOURCE_TELL, and ZIP_SOURCE_SUPPORTS.
read/write source	Same as previous, but additionally allowing writing (also for writable zip archives). Must additionally support ZIP_SOURCE_BEGIN_WRITE, ZIP_SOURCE_COMMIT_WRITE, ZIP_SOURCE_ROLLBACK_WRITE, ZIP_SOURCE_SEEK_WRITE, ZIP_SOURCE_TELL_WRITE, and ZIP_SOURCE_REMOVE. On top of the above, supporting the pseudo-command ZIP_SOURCE_SUPPORTS_REOPEN allows calling **zip_source_open**() again after calling **zip_source_close**().

ZIP_SOURCE_ACCEPT_EMPTY
Return 1 if an empty source should be accepted as a valid zip archive. This is the default if this command is not supported by a source. File system backed sources should return 0.

ZIP_SOURCE_BEGIN_WRITE
Prepare the source for writing. Use this to create any temporary file(s).

ZIP_SOURCE_BEGIN_WRITE_CLONING
Prepare the source for writing, keeping the first *len* bytes of the original file. Only implement this command if it is more efficient than copying the data, and if it does not destructively overwrite the original file (you still have to be able to execute ZIP_SOURCE_ROLLBACK_WRITE).

The next write should happen at byte *offset*.

ZIP_SOURCE_CLOSE
Reading is done.

ZIP_SOURCE_COMMIT_WRITE
Finish writing to the source. Replace the original data with the newly written data. Clean up temporary files or internal buffers. Subsequently opening and reading from the source should return the newly written data.

ZIP_SOURCE_ERROR
Get error information. *data* points to an array of two ints, which should be filled with the libzip error code and the corresponding system error code for the error that occurred. See zip_errors(3) for details on the error codes. If the source stores error information in a zip_error_t, use zip_error_to_data(3) and return its return value. Otherwise, return 2 * sizeof(int).

ZIP_SOURCE_FREE
Clean up and free all resources, including *userdata*. The callback function will not be called again.

ZIP_SOURCE_GET_FILE_ATTRIBUTES
Provide information about various data. Then the data should be put in the appropriate entry in the passed *zip_file_attributes_t* argument, and the appropriate ZIP_FILE_ATTRIBUTES_* value must be or'ed into the *valid* member to denote that the corresponding data has been provided. A *zip_file_attributes_t* structure can be initialized using zip_file_attributes_init(3).

ASCII mode
If a file is a plaintext file in ASCII. Can be used by extraction tools to automatically convert line endings (part of the internal file attributes). Member *ascii*, flag ZIP_FILE_ATTRIBUTES_ASCII.

General Purpose Bit Flags (limited to Compression Flags)
The general purpose bit flag in the zip in the local and central directory headers contain information about the compression method. Member *general_purpose_bit_flags* and *general_purpose_bit_mask* to denote which members have been set; flag ZIP_FILE_ATTRIBUTES_GENERAL_PURPOSE_BIT_FLAGS.

External File Attributes
The external file attributes (usually operating system-specific). Member *external_file_attributes*, flag ZIP_FILE_ATTRIBUTES_EXTERNAL_FILE_ATTRIBUTES.

Version Needed
A minimum version needed required to unpack this entry (in the usual "major * 10 + minor" format). Member *version_needed*, flag ZIP_FILE_ATTRIBUTES_VERSION_NEEDED.

Operating System
One of the operating systems as defined by the ZIP_OPSYS_* variables (see *zip.h*). This value affects the interpretation of the external file attributes. Member *host_system*, flag ZIP_FILE_ATTRIBUTES_HOST_SYSTEM.

ZIP_SOURCE_OPEN
Prepare for reading.

ZIP_SOURCE_READ
Read data into the buffer *data* of size *len*. Return the number of bytes placed into *data* on success, and

zero for end-of-file.

ZIP_SOURCE_REMOVE
Remove the underlying file. This is called if a zip archive is empty when closed.

ZIP_SOURCE_ROLLBACK_WRITE
Abort writing to the source. Discard written data. Clean up temporary files or internal buffers. Subsequently opening and reading from the source should return the original data.

ZIP_SOURCE_SEEK
Specify position to read next byte from, like fseek(3). Use ZIP_SOURCE_GET_ARGS(3) to decode the arguments into the following struct:

```
struct zip_source_args_seek {
    zip_int64_t offset;
    int whence;
};
```

If the size of the source's data is known, use zip_source_seek_compute_offset(3) to validate the arguments and compute the new offset.

ZIP_SOURCE_SEEK_WRITE
Specify position to write next byte to, like fseek(3). See ZIP_SOURCE_SEEK for details.

ZIP_SOURCE_STAT
Get meta information for the input data. *data* points to an allocated *struct zip_stat*, which should be initialized using zip_stat_init(3) and then filled in.

For uncompressed, unencrypted data, all information is optional. However, fill in as much information as is readily available.

If the data is compressed, ZIP_STAT_COMP_METHOD, ZIP_STAT_SIZE, and ZIP_STAT_CRC must be filled in.

If the data is encrypted, ZIP_STAT_ENCRYPTION_METHOD, ZIP_STAT_COMP_METHOD, ZIP_STAT_SIZE, and ZIP_STAT_CRC must be filled in.

Information only available after the source has been read (e.g., size) can be omitted in an earlier call. *NOTE*: **zip_source_function**() may be called with this argument even after being called with ZIP_SOURCE_CLOSE.

Return sizeof(struct zip_stat) on success.

ZIP_SOURCE_SUPPORTS
Return bitmap specifying which commands are supported. Use zip_source_make_command_bitmap(3). If this command is not implemented, the source is assumed to be a read source without seek support.

ZIP_SOURCE_TELL
Return the current read offset in the source, like ftell(3).

ZIP_SOURCE_TELL_WRITE
Return the current write offset in the source, like ftell(3).

ZIP_SOURCE_WRITE
Write data to the source. Return number of bytes written.

ZIP_SOURCE_SUPPORTS_REOPEN
This command is never actually invoked, support for it signals the ability to handle multiple open/read/close cycles.

Return Values

Commands should return -1 on error. ZIP_SOURCE_ERROR will be called to retrieve the error code. On success, commands return 0, unless specified otherwise in the description above.

Calling Conventions

The library will always issue ZIP_SOURCE_OPEN before issuing ZIP_SOURCE_READ, ZIP_SOURCE_SEEK, or ZIP_SOURCE_TELL. When it no longer wishes to read from this source, it will issue ZIP_SOURCE_CLOSE. If the library wishes to read the data again, it will issue ZIP_SOURCE_OPEN a second time. If the function is unable to provide the data again, it should return -1.

ZIP_SOURCE_BEGIN_WRITE or ZIP_SOURCE_BEGIN_WRITE_CLONING will be called before ZIP_SOURCE_WRITE, ZIP_SOURCE_SEEK_WRITE, or ZIP_SOURCE_TELL_WRITE. When writing is complete, either ZIP_SOURCE_COMMIT_WRITE or ZIP_SOURCE_ROLLBACK_WRITE will be called.

ZIP_SOURCE_ACCEPT_EMPTY, ZIP_SOURCE_GET_FILE_ATTRIBUTES, and ZIP_SOURCE_STAT can be issued at any time.

ZIP_SOURCE_ERROR will only be issued in response to the function returning -1.

ZIP_SOURCE_FREE will be the last command issued; if ZIP_SOURCE_OPEN was called and succeeded, ZIP_SOURCE_CLOSE will be called before ZIP_SOURCE_FREE, and similarly for ZIP_SOURCE_BEGIN_WRITE or ZIP_SOURCE_BEGIN_WRITE_CLONING and ZIP_SOURCE_COMMIT_WRITE or ZIP_SOURCE_ROLLBACK_WRITE.

RETURN VALUES

Upon successful completion, the created source is returned. Otherwise, NULL is returned and the error code in *archive* or *error* is set to indicate the error (unless it is NULL).

ERRORS

zip_source_function() fails if:

[ZIP_ER_MEMORY]
Required memory could not be allocated.

SEE ALSO

libzip(3), zip_file_add(3), zip_file_attributes_init(3), zip_file_replace(3), zip_source(3), zip_stat_init(3)

HISTORY

zip_source_function() and **zip_source_function_create**() were added in libzip 1.0.

AUTHORS

Dieter Baron <*dillo@nih.at*> and Thomas Klausner <*tk@giga.or.at*>