NAME

archive_write_open, **archive_write_open2**, **archive_write_open_fd**, **archive_write_open_FILE**, **archive_write_open_filename**, **archive_write_open_memory** - functions for creating archives

LIBRARY

Streaming Archive Library (libarchive, -larchive)

SYNOPSIS

#include <archive.h>

int
archive_write_open(*struct archive *, void *client_data, archive_open_callback *, archive_write_callback *, archive_close_callback **);

int
archive_write_open2(*struct archive *, void *client_data, archive_open_callback *, archive_write_callback *, archive_close_callback *, archive_free_callback **);

int
archive_write_open_fd(*struct archive *, int fd*);

int
archive_write_open_FILE(*struct archive *, FILE *file*);

int
archive_write_open_filename(*struct archive *, const char *filename*);

int
archive_write_open_memory(*struct archive *, void *buffer, size_t bufferSize, size_t *outUsed*);

DESCRIPTION

archive_write_open()

Freeze the settings, open the archive, and prepare for writing entries. This is the most generic form of this function, which accepts pointers to three callback functions which will be invoked by the compression layer to write the constructed archive. This does not alter the default archive padding.

archive_write_open2()

Same as **archive_write_open**() with an additional fourth free callback. This function should be preferred to **archive_write_open**().

archive_write_open_fd()
A convenience form of **archive_write_open**() that accepts a file descriptor. The **archive_write_open_fd**() function is safe for use with tape drives or other block-oriented devices.

archive_write_open_FILE()
A convenience form of **archive_write_open**() that accepts a *FILE* * pointer. Note that **archive_write_open_FILE**() is not safe for writing to tape drives or other devices that require correct blocking.

archive_write_open_file()
A deprecated synonym for **archive_write_open_filename**().

archive_write_open_filename()
A convenience form of **archive_write_open**() that accepts a filename. A NULL argument indicates that the output should be written to standard output; an argument of "-" will open a file with that name. If you have not invoked **archive_write_set_bytes_in_last_block**(), then **archive_write_open_filename**() will adjust the last-block padding depending on the file: it will enable padding when writing to standard output or to a character or block device node, it will disable padding otherwise. You can override this by manually invoking **archive_write_set_bytes_in_last_block**() before calling **archive_write_open2**(). The **archive_write_open_filename**() function is safe for use with tape drives or other block-oriented devices.

archive_write_open_memory()
A convenience form of **archive_write_open2**() that accepts a pointer to a block of memory that will receive the archive. The final *size_t* * argument points to a variable that will be updated after each write to reflect how much of the buffer is currently in use. You should be careful to ensure that this variable remains allocated until after the archive is closed. This function will disable padding unless you have specifically set the block size.

More information about the *struct archive* object and the overall design of the library can be found in the libarchive(3) overview.

Note that the convenience forms above vary in how they block the output. See archive_write_blocksize(3) if you need to control the block size used for writes or the end-of-file padding behavior.

CLIENT CALLBACKS

To use this library, you will need to define and register callback functions that will be invoked to write data to the resulting archive. These functions are registered by calling **archive_write_open2**():

typedef int **archive_open_callback**(*struct archive *, void *client_data*)

The open callback is invoked by **archive_write_open**(). It should return **ARCHIVE_OK** if the underlying file or data source is successfully opened. If the open fails, it should call **archive_set_error**() to register an error code and message and return **ARCHIVE_FATAL**. Please note that if open fails, close is not called and resources must be freed inside the open callback or with the free callback.

typedef la_ssize_t **archive_write_callback**(*struct archive *, void *client_data, const void *buffer, size_t length*)

The write callback is invoked whenever the library needs to write raw bytes to the archive. For correct blocking, each call to the write callback function should translate into a single write(2) system call. This is especially critical when writing archives to tape drives. On success, the write callback should return the number of bytes actually written. On error, the callback should invoke **archive_set_error**() to register an error code and message and return -1.

typedef int **archive_close_callback**(*struct archive *, void *client_data*)

The close callback is invoked by archive_close when the archive processing is complete. If the open callback fails, the close callback is not invoked. The callback should return **ARCHIVE_OK** on success. On failure, the callback should invoke **archive_set_error**() to register an error code and message and return **ARCHIVE_FATAL**.

typedef int **archive_free_callback**(*struct archive *, void *client_data*)

The free callback is always invoked on archive_free. The return code of this callback is not processed.

Note that if the client-provided write callback function returns a non-zero value, that error will be propagated back to the caller through whatever API function resulted in that call, which may include **archive_write_header**(), **archive_write_data**(), **archive_write_close**(), **archive_write_finish**(), or **archive_write_free**(). The client callback can call **archive_set_error**() to provide values that can then be retrieved by **archive_errno**() and **archive_error_string**().

RETURN VALUES

These functions return **ARCHIVE_OK** on success, or **ARCHIVE_FATAL**.

ERRORS

Detailed error codes and textual descriptions are available from the **archive_errno**() and **archive_error_string**() functions.

SEE ALSO

tar(1), archive_write(3), archive_write_blocksize(3), archive_write_filter(3), archive_write_format(3), archive_write_new(3), archive_write_set_options(3), libarchive(3), cpio(5), mtree(5), tar(5)