NAME

  archive_clear_error,   **archive_compression**,   **archive_compression_name**,
  archive_copy_error, **archive_errno**, **archive_error_string**, **archive_file_count**,
  archive_filter_code,   **archive_filter_count**,   **archive_filter_name**,
  archive_format, **archive_format_name**, **archive_position**, **archive_set_error**
  libarchive utility functions

LIBRARY

  Streaming Archive Library (libarchive, -larchive)

SYNOPSIS

  #include <archive.h>

  void
  archive_clear_error(*struct archive **);

  int
  archive_compression(*struct archive **);

  *const char **
  archive_compression_name(*struct archive **);

  void
  archive_copy_error(*struct archive **, *struct archive **);

  int
  archive_errno(*struct archive **);

  *const char **
  archive_error_string(*struct archive **);

  int
  archive_file_count(*struct archive **);

  int
  archive_filter_code(*struct archive **, *int*);

  int
  archive_filter_count(*struct archive **, *int*);

  *const char **
  archive_filter_name(*struct archive **, *int*);

  int
  archive_format(*struct archive **);

  *const char **
  archive_format_name(*struct archive **);

  int64_t
  archive_position(*struct archive **, *int*);

  void
  archive_set_error(*struct archive **, *int error_code*, *const char *fmt*, *...*);

DESCRIPTION

  These functions provide access to various information about the archive object used in the
  libarchive(3) library.

archive_clear_error()
        Clears any error information left over from a previous call. Not generally used in client code.

archive_compression()
        Synonym for **archive_filter_code(a,** *(0)*).

archive_compression_name()
        Synonym for **archive_filter_name(a,** *(0)*).

archive_copy_error()
        Copies error information from one archive to another.

archive_errno()
        Returns a numeric error code (see errno(2)) indicating the reason for the most recent error
        return. Note that this can not be reliably used to detect whether an error has occurred. It should be
        used only after another library function has returned an error status.

archive_error_string()
        Returns a textual error message suitable for display. The error message here is usually more spe-
        cific than that obtained from passing the result of **archive_errno**() to strerror(3).

archive_file_count()
        Returns a count of the number of files processed by this archive. The count is incremented
        by calls to archive_write_header(3) or archive_read_next_header(3).

archive_filter_code()
        Returns a numeric code identifying the indicated filter. See **archive_filter_count**() for
        details of the numbering.

archive_filter_count()
        Returns the number of filters in the current pipeline. For read archive handles, these filters are
        added automatically by the automatic format detection. For write archive handles, these filters are
        added by calls to the various **archive_write_add_filter_XXX**() functions. Filters in the
        resulting pipeline are numbered so that filter 0 is the filter closest to the format. As a con-
        venience, functions that expect a filter number will accept -1 as a synonym for the highest-num-
        bered filter.

        For example, when reading a uuencoded gzipped tar archive, there are three filters: filter 0 is the
        gunzip filter, filter 1 is the uudecode, and filter 2 is the pseudo-filter that wraps the archive
        read functions. In this case, requesting **archive_position(a,** *(−1)*) would be a synonym for
        archive_position(a, *(2)*) which would return the number of bytes currently read from the
        archive, while **archive_position(a,** *(1)*) would return the number of bytes after uudecod-
        ing, and **archive_position(a,** *(0)*) would return the number of bytes after decompression.

archive_filter_name()
        Returns a textual name identifying the indicated filter. See **archive_filter_count**() for
        details of the numbering.

archive_format()
        Returns a numeric code indicating the format of the current archive entry. This value is set by a
        successful call to **archive_read_next_header**(). Note that it is common for this to
        change from entry to entry. For example, a tar archive might have several entries that utilize GNU
        tar extensions and several entries that do not. These entries will have different format codes.

archive_format_name()
        A textual description of the format of the current entry.

archive_position()
        Returns the number of bytes read from or written to the indicated filter. In particular,
        archive_position(a, *(0)*) returns the number of bytes read or written by the format han-
        dler, while **archive_position(a,** *(−1)*) returns the number of bytes read or written to the ar-
        chive. See **archive_filter_count**() for details of the numbering here.

archive_set_error()

Sets the numeric error code and error description that will be returned by **archive_errno**() and
archive_error_string(). This function should be used within I/O callbacks to set system-
specific error codes and error descriptions. This function accepts a printf-like format string and
arguments. However, you should be careful to use only the following printf format specifiers:
%c , %d , %jd , %jo , %ju , %jx , %ld , %lo , %lu , %lx , %o , %u , %s ,
%x , %% . Field-width specifiers and other printf features are not uniformly supported and
should not be used.

SEE ALSO

archive_read(3), archive_write(3), libarchive(3), printf(3)

HISTORY

The **libarchive** library first appeared in FreeBSD 5.3.

AUTHORS

The **libarchive** library was written by Tim Kientzle <kientzle@acm.org>.