NAME

archive_read_support_format_7zip, archive_read_support_format_all, archive_read_support_format_ar, archive_read_support_format_by_code, archive_read_support_format_cab, archive_read_support_format_cpio, archive_read_support_format_empty, archive_read_support_format_iso9660, archive_read_support_format_lha, archive_read_support_format_mtree, archive_read_support_format_rar, archive_read_support_format_rar5, archive_read_support_format_raw, archive_read_support_format_tar, archive_read_support_format_warc, archive_read_support_format_xar, archive_read_support_format_zip — functions for reading streaming archives

LIBRARY

Streaming Archive Library (libarchive, -larchive)

SYNOPSIS

```
#include <archive.h>

int
archive_read_support_format_7zip(struct archive *);

int
archive_read_support_format_all(struct archive *);

int
archive_read_support_format_ar(struct archive *);

int
archive_read_support_format_by_code(struct archive *, int);

int
archive_read_support_format_cab(struct archive *);

int
archive_read_support_format_cpio(struct archive *);

int
archive_read_support_format_empty(struct archive *);

int
archive_read_support_format_iso9660(struct archive *);

int
archive_read_support_format_lha(struct archive *);

int
archive_read_support_format_mtree(struct archive *);

int
archive_read_support_format_rar(struct archive *);

int
archive_read_support_format_rar5(struct archive *);

int
archive_read_support_format_raw(struct archive *);

int
archive_read_support_format_tar(struct archive *);

int
archive_read_support_format_warc(struct archive *);

int
archive_read_support_format_xar(struct archive *);
```

int
archive_read_support_format_zip(*struct archive* *);

DESCRIPTION

archive_read_support_format_7zip(), **archive_read_support_format_ar**(), **archive_read_support_format_cab**(), **archive_read_support_format_cpio**(), **archive_read_support_format_iso9660**(), **archive_read_support_format_lha**(), **archive_read_support_format_mtree**(), **archive_read_support_format_rar**(), **archive_read_support_format_rar5**(), **archive_read_support_format_raw**(), **archive_read_support_format_tar**(), **archive_read_support_format_warc**(), **archive_read_support_format_xar**(), **archive_read_support_format_zip**()
Enables support---including auto-detection code---for the specified archive format. For example, **archive_read_support_format_tar**() enables support for a variety of standard tar formats, old-style tar, ustar, pax interchange format, and many common variants. **archive_read_support_format_zip**() enables support for both the streaming and the seeking zip readers, which can separately be enabled by respectively **archive_read_support_format_zip_streamable**() and **archive_read_support_format_zip_seekable**()

archive_read_support_format_all()
Enables support for all available formats except the "raw" format (see below).

archive_read_support_format_by_code()
Enables a single format specified by the format code. This can be useful when reading a single archive twice; use **archive_format**() after reading the first time and pass the resulting code to this function to selectively enable only the necessary format support. Note: In statically-linked executables, this will cause your program to include support for every format. If executable size is a concern, you may wish to avoid using this function.

archive_read_support_format_empty()
Enables support for treating empty files as empty archives. Because empty files are valid for several different formats, it is not possible to accurately determine a format for an empty file based purely on contents. So empty files are treated by libarchive as a distinct format.

archive_read_support_format_raw()
The "raw" format handler allows libarchive to be used to read arbitrary data. It treats any data stream as an archive with a single entry. The pathname of this entry is "data"; all other entry fields are unset. This is not enabled by **archive_read_support_format_all**() in order to avoid erroneous handling of damaged archives.

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_read_data*(3), *archive_read_filter*(3), *archive_read_set_options*(3), *archive_util*(3), *libarchive*(3), *tar*(5)

BUGS

Many traditional archiver programs treat empty files as valid empty archives. For example, many implementations of *tar*(1) allow you to append entries to an empty file. Of course, it is impossible to determine the format of an empty file by inspecting the contents, so this library treats empty files as having a special "empty" format.

Using the “raw” handler together with any other handler will often work but can produce surprising results.