NAME

archive_write_set_filter_option, archive_write_set_format_option, archive_write_set_option, archive_write_set_options — functions controlling options for writing archives

LIBRARY

Streaming Archive Library (libarchive, -larchive)

SYNOPSIS

```
int
archive_write_set_filter_option(struct archive *, const char *module,
    const char *option, const char *value);

int
archive_write_set_format_option(struct archive *, const char *module,
    const char *option, const char *value);

int
archive_write_set_option(struct archive *, const char *module,
    const char *option, const char *value);

int
archive_write_set_options(struct archive *, const char *options);
```

DESCRIPTION

These functions provide a way for libarchive clients to configure specific write modules.

archive_write_set_filter_option(), **archive_write_set_format_option**()
: Specifies an option that will be passed to the currently-registered filters (including decompression filters) or format readers.

  If *option* and *value* are both NULL, these functions will do nothing and ARCHIVE_OK will be returned. If *option* is NULL but *value* is not, these functions will do nothing and ARCHIVE_FAILED will be returned.

  If *module* is not NULL, *option* and *value* will be provided to the filter or reader named *module*. The return value will be either ARCHIVE_OK if the option was successfully handled or ARCHIVE_WARN if the option was unrecognized by the module or could otherwise not be handled. If there is no such module, ARCHIVE_FAILED will be returned.

  If *module* is NULL, *option* and *value* will be provided to every registered module. If any module returns ARCHIVE_FATAL, this value will be returned immediately. Otherwise, ARCHIVE_OK will be returned if any module accepts the option, and ARCHIVE_FAILED in all other cases.

archive_write_set_option()
: Calls **archive_write_set_format_option**(), then **archive_write_set_filter_option**(). If either function returns ARCHIVE_FATAL, ARCHIVE_FATAL will be returned immediately. Otherwise, the greater of the two values will be returned.

archive_write_set_options()
: *options* is a comma-separated list of options. If *options* is NULL or empty, ARCHIVE_OK will be returned immediately.

  Individual options have one of the following forms:

  option=value
  : The option/value pair will be provided to every module. Modules that do not accept an option with this name will ignore it.

  option The option will be provided to every module with a value of "1".

!option
: The option will be provided to every module with a NULL value.

module:option=value, *module:option*, *module:!option*
: As above, but the corresponding option and value will be provided only to modules whose name matches *module*.

OPTIONS

Filter b64encode

`mode` The value is interpreted as octal digits specifying the file mode.

`name` The value specifies the file name.

Filter bzip2

`compression-level`
: The value is interpreted as a decimal integer specifying the bzip2 compression level. Supported values are from 1 to 9.

Filter gzip

`compression-level`
: The value is interpreted as a decimal integer specifying the gzip compression level. Supported values are from 0 to 9.

`timestamp`
: Store timestamp. This is enabled by default.

Filter lrzip

`compression=`*type*
: Use *type* as compression method. Supported values are “bzip2”, “gzipi”, “lzo” (ultra fast), and “zpaq” (best, extremely slow).

`compression-level`
: The value is interpreted as a decimal integer specifying the lrzip compression level. Supported values are from 1 to 9.

Filter lz4

`compression-level`
: The value is interpreted as a decimal integer specifying the lz4 compression level. Supported values are from 0 to 9.

`stream-checksum`
: Enable stream checksum. This is enabled by default.

`block-checksum`
: Enable block checksum. This is disabled by default.

`block-size`
: The value is interpreted as a decimal integer specifying the lz4 compression block size. Supported values are from 4 to 7 (default).

`block-dependence`
: Use the previous block of the block being compressed for a compression dictionary to improve compression ratio. This is disabled by default.

Filter lzop

`compression-level`
: The value is interpreted as a decimal integer specifying the lzop compression level. Supported values are from 1 to 9.

Filter uuencode

`mode` The value is interpreted as octal digits specifying the file mode.

`name` The value specifies the file name.

Filter xz

`compression-level`
: The value is interpreted as a decimal integer specifying the compression level. Supported values are from 0 to 9.

`threads`
: The value is interpreted as a decimal integer specifying the number of threads for multi-threaded lzma compression. If supported, the default value is read from **`lzma_cputhreads`**().

Filter zstd

`compression-level`
: The value is interpreted as a decimal integer specifying the compression level. Supported values depend on the library version, common values are from 1 to 22.

`long`
: Enables long distance matching. The value is interpreted as a decimal integer specifying log2 window size in bytes. Values from 10 to 30 for 32 bit, or 31 for 64 bit, are supported.

`threads`
: The value is interpreted as a decimal integer specifying the number of threads for multi-threaded zstd compression. If set to 0, zstd will attempt to detect and use the number of physical CPU cores.

Format 7zip

`compression`
: The value is one of “store”, “copy”, “deflate”, “bzip2”, “lzma1”, “lzma2” or “ppmd” to indicate how the following entries should be compressed. The values “store” and “copy” are synonyms. Note that this setting is ignored for directories, symbolic links, and other special entries.

`compression-level`
: The value is interpreted as a decimal integer specifying the compression level. Values between 0 and 9 are supported, with the exception of bzip2 which only supports values between 1 and 9. The interpretation of the compression level depends on the chosen compression method.

Format bin

`hdrcharset`
: The value is used as a character set name that will be used when translating file names.

Format gnutar

`hdrcharset`
: The value is used as a character set name that will be used when translating file, group and user names.

Format iso9660 - volume metadata

These options are used to set standard ISO9660 metadata.

`abstract-file=`*`filename`*
: The file with the specified name will be identified in the ISO9660 metadata as holding the abstract for this volume. Default: none.

`application-id=`*`filename`*
: The file with the specified name will be identified in the ISO9660 metadata as holding the application identifier for this volume. Default: none.

`biblio-file=`*`filename`*
: The file with the specified name will be identified in the ISO9660 metadata as holding the bibliography for this volume. Default: none.

`copyright-file=`*`filename`*
: The file with the specified name will be identified in the ISO9660 metadata as holding the copyright for this volume. Default: none.

`publisher=`*`filename`*
: The file with the specified name will be identified in the ISO9660 metadata as holding the publisher information for this volume. Default: none.

`volume-id=`*`string`*
: The specified string will be used as the Volume Identifier in the ISO9660 metadata. It is limited to 32 bytes. Default: none.

Format iso9660 - boot support

These options are used to make an ISO9660 image that can be directly booted on various systems.

`boot=`*`filename`*

The file matching this name will be used as the El Torito boot image file.

`boot-catalog=`*`name`*

The name that will be used for the El Torito boot catalog. Default: *`boot.catalog`*

`boot-info-table`

The boot image file provided by the `boot=`*`filename`* option will be edited with appropriate boot information in bytes 8 through 64. Default: disabled

`boot-load-seg=`*`hexadecimal-number`*

The load segment for a no-emulation boot image.

`boot-load-size=`*`decimal-number`*

The number of "virtual" 512-byte sectors to be loaded from a no-emulation boot image. Some very old BIOSes can only load very small images, setting this value to 4 will often allow such BIOSes to load the first part of the boot image (which will then need to be intelligent enough to load the rest of itself). This should not be needed unless you are trying to support systems with very old BIOSes. This defaults to the full size of the image.

`boot-type=`*`value`*

Specifies the boot semantics used by the El Torito boot image: If the *`value`* is `fd`, then the boot image is assumed to be a bootable floppy image. If the *`value`* is `hd`, then the boot image is assumed to be a bootable hard disk image. If the *`value`* is `no-emulation`, the boot image is used without floppy or hard disk emulation. If the boot image is exactly 1.2MB, 1.44MB, or 2.88MB, then the default is `fd`, otherwise the default is `no-emulation`.

Format iso9660 - filename and size extensions

Various extensions to the base ISO9660 format.

`allow-ldots`

If enabled, allows filenames to begin with a leading period. If disabled, filenames that begin with a leading period will have that period replaced by an underscore character in the standard ISO9660 namespace. This does not impact names stored in the Rockridge or Joliet extension area. Default: disabled.

`allow-lowercase`

If enabled, allows filenames to contain lowercase characters. If disabled, filenames will be forced to uppercase. This does not impact names stored in the Rockridge or Joliet extension area. Default: disabled.

`allow-multidot`

If enabled, allows filenames to contain multiple period characters, in violation of the ISO9660 specification. If disabled, additional periods will be converted to underscore characters. This does not impact names stored in the Rockridge or Joliet extension area. Default: disabled.

`allow-period`

If enabled, allows filenames to contain trailing period characters, in violation of the ISO9660 specification. If disabled, trailing periods will be converted to underscore characters. This does not impact names stored in the Rockridge or Joliet extension area. Default: disabled.

`allow-pvd-lowercase`

If enabled, the Primary Volume Descriptor may contain lowercase ASCII characters, in violation of the ISO9660 specification. If disabled, characters will be converted to uppercase ASCII. Default: disabled.

`allow-sharp-tilde`

If enabled, sharp and tilde characters will be permitted in filenames, in violation if the ISO9660 specification. If disabled, such characters will be converted to underscore

characters. Default: disabled.

`allow-vernum`
: If enabled, version numbers will be included with files. If disabled, version numbers will be suppressed, in violation of the ISO9660 standard. This does not impact names stored in the Rockridge or Joliet extension area. Default: enabled.

`iso-level`
: This enables support for file size and file name extensions in the core ISO9660 area. The name extensions specified here do not affect the names stored in the Rockridge or Joliet extension areas.

`iso-level=1`
: The most compliant form of ISO9660 image. Filenames are limited to 8.3 uppercase format, directory names are limited to 8 uppercase characters, files are limited to 4 GiB, the complete ISO9660 image cannot exceed 4 GiB.

`iso-level=2`
: Filenames are limited to 30 uppercase characters with a 30-character extension, directory names are limited to 30 characters, files are limited to 4 GiB.

`iso-level=3`
: As with `iso-level=2`, except that files may exceed 4 GiB.

`iso-level=4`
: As with `iso-level=3`, except that filenames may be up to 193 characters and may include arbitrary 8-bit characters.

`joliet` Microsoft's Joliet extensions store a completely separate set of directory information about each file. In particular, this information includes Unicode filenames of up to 255 characters. Default: enabled.

`limit-depth`
: If enabled, libarchive will use directory relocation records to ensure that no pathname exceeds the ISO9660 limit of 8 directory levels. If disabled, no relocation will occur. Default: enabled.

`limit-dirs`
: If enabled, libarchive will cause an error if there are more than 65536 directories. If disabled, there is no limit on the number of directories. Default: enabled

`pad` If enabled, 300 kiB of zero bytes will be appended to the end of the archive. Default: enabled

`relaxed-filenames`
: If enabled, all 7-bit ASCII characters are permitted in filenames (except lowercase characters unless `allow-lowercase` is also specified). This violates ISO9660 standards. This does not impact names stored in the Rockridge or Joliet extension area. Default: disabled.

`rockridge`
: The Rockridge extensions store an additional set of POSIX-style file information with each file, including mtime, atime, ctime, permissions, and long filenames with arbitrary 8-bit characters. These extensions also support symbolic links and other POSIX file types. Default: enabled.

Format iso9660 - zisofs support

The zisofs extensions permit each file to be independently compressed using a gzip-compatible compression. This can provide significant size savings, but requires the reading system to have support for these extensions. These extensions are disabled by default.

`compression-level`=number
: The compression level used by the deflate compressor. Ranges from 0 (least effort) to 9 (most effort). Default: 6

`zisofs` Synonym for `zisofs=direct`.

`zisofs=direct`
: Compress each file in the archive. Unlike `zisofs=indirect`, this is handled entirely within libarchive and does not require a separate utility. For best results,

libarchive tests each file and will store the file uncompressed if the compression does not actually save any space. In particular, files under 2k will never be compressed. Note that boot image files are never compressed.

`zisofs=indirect`
: Recognizes files that have already been compressed with the `mkzftree` utility and sets up the necessary file metadata so that readers will correctly identify these as zisofs-compressed files.

`zisofs-exclude=`*`filename`*
: Specifies a filename that should not be compressed when using `zisofs=direct`. This option can be provided multiple times to suppress compression on many files.

Format mtree

`cksum`, `device`, `flags`, `gid`, `gname`, `indent`, `link`, `md5`, `mode`, `nlink`, `rmd160`, `sha1`, `sha256`, `sha384`, `sha512`, `size`, `time`, `uid`, `uname`
: Enable a particular keyword in the mtree output. Prefix with an exclamation mark to disable the corresponding keyword. The default is equivalent to "device, flags, gid, gname, link, mode, nlink, size, time, type, uid, uname".

`all`
: Enables all of the above keywords.

`use-set`
: Enables generation of `/set` lines that specify default values for the following files and/or directories.

`indent`
: XXX needs explanation XXX

Format newc

`hdrcharset`
: The value is used as a character set name that will be used when translating file names.

Format odc

`hdrcharset`
: The value is used as a character set name that will be used when translating file names.

Format pwb

`hdrcharset`
: The value is used as a character set name that will be used when translating file names.

Format pax

`hdrcharset`
: The value is used as a character set name that will be used when translating file, group and user names. The value is one of "BINARY" or "UTF-8". With "BINARY" there is no character conversion, with "UTF-8" names are converted to UTF-8.

`xattrheader`
: When storing extended attributes, this option configures which headers should be written. The value is one of "all", "LIBARCHIVE", or "SCHILY". By default, both "LIBARCHIVE.xattr" and "SCHILY.xattr" headers are written.

Format ustar

`hdrcharset`
: The value is used as a character set name that will be used when translating file, group and user names.

Format v7tar

`hdrcharset`
: The value is used as a character set name that will be used when translating file, group and user names.

Format warc

`omit-warcinfo`
: Set to "true" to disable output of the warcinfo record.

Format xar

`checksum=`*`type`*
: Use *`type`* as file checksum method. Supported values are "none", "md5", and "sha1" (default).

`compression=`*`type`*
: Use *`type`* as compression method. Supported values are “none”, “bzip2”, “gzip” (default), “lzma” and “xz”.

`compression_level`
: The value is a decimal integer from 1 to 9 specifying the compression level.

`toc-checksum=`*`type`*
: Use *`type`* as table of contents checksum method. Supported values are “none”, “md5” and “sha1” (default).

Format zip

`compression`
: The value is either “store” or “deflate” to indicate how the following entries should be compressed. Note that this setting is ignored for directories, symbolic links, and other special entries.

`compression-level`
: The value is interpreted as a decimal integer specifying the compression level. Values between 0 and 9 are supported. A compression level of 0 switches the compression method to “store”, other values will enable “deflate” compression with the given level.

`encryption`
: Enable encryption using traditional zip encryption.

`encryption=`*`type`*
: Use *`type`* as encryption type. Supported values are “zipcrypt” (traditional zip encryption), “aes128” (WinZip AES-128 encryption) and “aes256” (WinZip AES-256 encryption).

`experimental`
: This boolean option enables or disables experimental Zip features that may not be compatible with other Zip implementations.

`fakecrc32`
: This boolean option disables CRC calculations. All CRC fields are set to zero. It should not be used except for testing purposes.

`hdrcharset`
: The value is used as a character set name that will be used when translating file names.

`zip64` Zip64 extensions provide additional file size information for entries larger than 4 GiB. They also provide extended file offset and archive size information when archives exceed 4 GiB. By default, the Zip writer selectively enables these extensions only as needed. In particular, if the file size is unknown, the Zip writer will include Zip64 extensions to guard against the possibility that the file might be larger than 4 GiB.

Setting this boolean option will force the writer to use Zip64 extensions even for small files that would not otherwise require them. This is primarily useful for testing.

Disabling this option with `!zip64` will force the Zip writer to avoid Zip64 extensions: It will reject files with size greater than 4 GiB, it will reject any new entries once the total archive size reaches 4 GiB, and it will not use Zip64 extensions for files with unknown size. In particular, this can improve compatibility when generating archives where the entry sizes are not known in advance.

EXAMPLES

The following example creates an archive write handle to create a gzip-compressed ISO9660 format image. The two options here specify that the ISO9660 archive will use *`kernel.img`* as the boot image for El Torito booting, and that the gzip compressor should use the maximum compression level.

```
a = archive_write_new();
archive_write_add_filter_gzip(a);
archive_write_set_format_iso9660(a);
archive_write_set_options(a, "boot=kernel.img,compression=9");
archive_write_open_filename(a, filename, blocksize);
```

ERRORS

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

SEE ALSO

tar(1), *archive_read_set_options*(3), *archive_write*(3), *libarchive*(3)

HISTORY

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

AUTHORS

The options support for libarchive was originally implemented by Michihiro NAKAJIMA.

BUGS