

**NAME**

libarchive\_changes — changes in libarchive interface

**CHANGES IN LIBARCHIVE 3**

This page describes user-visible changes in libarchive3, and lists public functions and other symbols changed, deprecated or removed in libarchive3, along with their replacements if any.

**Multiple Filters**

Libarchive2 permitted a single (input or output) filter active on an archive. Libarchive3 extends this into a variable-length stack. Where **archive\_write\_set\_compression\_XXX()** would replace any existing filter, **archive\_write\_add\_filter\_XXX()** extends the write pipeline with another filter.

**Character Set Handling**

Libarchive2 assumed that the local platform uses Unicode as the native `wchar_t` encoding, which is true on Windows, modern Linux, and a few other systems, but is certainly not universal. As a result, pax format archives were written incorrectly on some systems, since pax format requires UTF-8 and libarchive 2 incorrectly assumed that `wchar_t` strings can be easily converted to UTF-8.

Libarchive3 uses the standard `iconv` library to convert between character sets and is introducing the notion of a “default character set for the archive”. To support this, `archive_entry` objects can now be bound to a particular archive when they are created. The automatic character set conversions performed by `archive_entry` objects when reading and writing filenames, usernames, and other strings will now use an appropriate default character set:

If the `archive_entry` object is bound to an archive, it will use the default character set for that archive.

The platform default character encoding (as returned by **nl\_langinfo(CHARSET)**) will be used if nothing else is specified.

Libarchive3 also introduces charset options to many of the archive readers and writers to control the character set that will be used for filenames written in those archives. When possible, this will be set automatically based on information in the archive itself. Combining this with the notion of a default character set for the archive should allow you to configure libarchive to read archives from other platforms and have the filenames and other information transparently converted to the character encoding suitable for your application.

**Prototype Changes**

These changes break binary compatibility; libarchive3 has a new shared library version to reflect these changes. The library now uses portable wide types such as `int64_t` instead of less-portable types such as `off_t`, `gid_t`, `uid_t`, and `ino_t`.

There are a few cases where these changes will affect your source code:

- In some cases, libarchive’s wider types will introduce the possibility of truncation: for example, on a system with a 16-bit `uid_t`, you risk having uid 65536 be truncated to uid 0, which can cause serious security problems.
- Typedef function pointer types will be incompatible. For example, if you define custom skip callbacks, you may have to use code similar to the following if you want to support building against libarchive2 and libarchive3:

```
#if ARCHIVE_VERSION_NUMBER < 3000000
typedef off_t myoff_t;
#else
typedef int64_t myoff_t;
#endif

myoff_t
my_skip_function(struct archive *a, void *v, myoff_t o)
{
    ... implementation ...
}
```

```
}
```

Affected functions:

- `archive_entry_gid()`, `archive_entry_set_gid()`
- `archive_entry_uid()`, `archive_entry_set_uid()`
- `archive_entry_ino()`, `archive_entry_set_ino()`
- `archive_read_data_block()`, `archive_write_data_block()`
- `archive_read_disk_gname()`, `archive_read_disk_uname()`
- `archive_read_disk_set_gname_lookup()`,  
`archive_read_disk_set_group_lookup()`,  
`archive_read_disk_set_uname_lookup()`,  
`archive_read_disk_set_user_lookup()`
- `archive_skip_callback()`
- `archive_read_extract_set_skip_file()`, `archive_write_disk_set_skip_file()`,  
`archive_write_set_skip_file()`
- `archive_write_disk_set_group_lookup()`,  
`archive_write_disk_set_user_lookup()`

Where these functions or their arguments took or returned `gid_t`, `ino_t`, `off_t`, or `uid_t` they now take or return `int64_t` or equivalent.

### Deprecated Symbols

Symbols deprecated in `libarchive3` will be removed in `libarchive4`. These symbols, along with their replacements if any, are listed below:

```
archive_position_compressed(), archive_position_uncompressed()
    archive_filter_bytes()

archive_compression()
    archive_filter_code()

archive_compression_name()
    archive_filter_name()

archive_read_finish(), archive_write_finish()
    archive_read_free(), archive_write_free()

archive_read_open_file(), archive_write_open_file()
    archive_read_open_filename(), archive_write_open_filename()

archive_read_support_compression_all()
    archive_read_support_filter_all()

archive_read_support_compression_bzip2()
    archive_read_support_filter_bzip2()

archive_read_support_compression_compress()
    archive_read_support_filter_compress()

archive_read_support_compression_gzip()
    archive_read_support_filter_gzip()

archive_read_support_compression_lzip()
    archive_read_support_filter_lzip()

archive_read_support_compression_lzma()
    archive_read_support_filter_lzma()

archive_read_support_compression_none()
    archive_read_support_filter_none()
```

```

archive_read_support_compression_program()
    archive_read_support_filter_program()

archive_read_support_compression_program_signature()
    archive_read_support_filter_program_signature()

archive_read_support_compression_rpm()
    archive_read_support_filter_rpm()

archive_read_support_compression_uu()
    archive_read_support_filter_uu()

archive_read_support_compression_xz()
    archive_read_support_filter_xz()

archive_write_set_compression_bzip2()
    archive_write_add_filter_bzip2()

archive_write_set_compression_compress()
    archive_write_add_filter_compress()

archive_write_set_compression_gzip()
    archive_write_add_filter_gzip()

archive_write_set_compression_lzip()
    archive_write_add_filter_lzip()

archive_write_set_compression_lzma()
    archive_write_add_filter_lzma()

archive_write_set_compression_none()
    archive_write_add_filter_none()

archive_write_set_compression_program()
    archive_write_add_filter_program()

archive_write_set_compression_filter()
    archive_write_add_filter_filter()

```

### Removed Symbols

These symbols, listed below along with their replacements if any, were deprecated in libarchive2, and are not part of libarchive3.

```

archive_api_feature()
    archive_version_number()

archive_api_version()
    archive_version_number()

archive_version()
    archive_version_string()

archive_version_stamp()
    archive_version_number()

archive_read_set_filter_options()
    archive_read_set_options() or archive_read_set_filter_option()

archive_read_set_format_options()
    archive_read_set_options() or archive_read_set_format_option()

archive_write_set_filter_options()
    archive_write_set_options() or archive_write_set_filter_option()

```

**archive\_write\_set\_format\_options()**  
**archive\_write\_set\_options()** or **archive\_write\_set\_format\_option()**

ARCHIVE\_API\_FEATURE  
ARCHIVE\_VERSION\_NUMBER

ARCHIVE\_API\_VERSION  
ARCHIVE\_VERSION\_NUMBER

ARCHIVE\_VERSION\_STAMP  
ARCHIVE\_VERSION\_NUMBER

ARCHIVE\_LIBRARY\_VERSION  
ARCHIVE\_VERSION\_STRING

ARCHIVE\_COMPRESSION\_NONE  
ARCHIVE\_FILTER\_NONE

ARCHIVE\_COMPRESSION\_GZIP  
ARCHIVE\_FILTER\_GZIP

ARCHIVE\_COMPRESSION\_BZIP2  
ARCHIVE\_FILTER\_BZIP2

ARCHIVE\_COMPRESSION\_COMPRESS  
ARCHIVE\_FILTER\_COMPRESS

ARCHIVE\_COMPRESSION\_PROGRAM  
ARCHIVE\_FILTER\_PROGRAM

ARCHIVE\_COMPRESSION\_LZMA  
ARCHIVE\_FILTER\_LZMA

ARCHIVE\_COMPRESSION\_XZ  
ARCHIVE\_FILTER\_XZ

ARCHIVE\_COMPRESSION\_UU  
ARCHIVE\_FILTER\_UU

ARCHIVE\_COMPRESSION\_RPM  
ARCHIVE\_FILTER\_RPM

ARCHIVE\_COMPRESSION\_LZIP  
ARCHIVE\_FILTER\_LZIP

ARCHIVE\_BYTES\_PER\_RECORD  
512

ARCHIVE\_DEFAULT\_BYTES\_PER\_BLOCK  
10240

**SEE ALSO**

*archive\_read(3)*, *archive\_read\_filter(3)*, *archive\_read\_format(3)*, *archive\_read\_set\_options(3)*,  
*archive\_util(3)*, *archive\_write(3)*, *archive\_write\_filter(3)*, *archive\_write\_format(3)*,  
*archive\_write\_set\_options(3)*, *libarchive(3)*