Building and installing a packaged release of jemalloc can be as simple as typing the following while in the root directory of the source tree:

```
./configure
make
make install
```

If building from unpackaged developer sources, the simplest command sequence that might work is:

```
./autogen.sh
make dist
make
make install
```

Note that documentation is not built by the default target because doing so would create a dependency on xsltproc in packaged releases, hence the requirement to either run 'make dist' or avoid installing docs via the various install_* targets documented below.

Advanced configuration

The 'configure' script supports numerous options that allow control of which functionality is enabled, where jemalloc is installed, etc. Optionally, pass any of the following arguments (not a definitive list) to 'configure':

- `--help`

  Print a definitive list of options.

- `--prefix=<install-root-dir>`

  Set the base directory in which to install. For example:

  ```
      ./configure --prefix=/usr/local
  ```

  will cause files to be installed into /usr/local/include, /usr/local/lib, and /usr/local/man.

- `--with-version=(<major>.<minor>.<bugfix>-<nrev>-g<gid>|VERSION)`

  The VERSION file is mandatory for successful configuration, and the following steps are taken to assure its presence:

  1. If --with-version=..--g is specified, generate VERSION using the specified value.
  2. If --with-version is not specified in either form and the source directory is inside a git repository, try to generate VERSION via 'git describe' invocations that pattern-match release tags.
  3. If VERSION is missing, generate it with a bogus version: 0.0.0-0-g0000000000000000000000000000000000000000

  Note that --with-version=VERSION bypasses (1) and (2), which simplifies VERSION configuration when embedding a jemalloc release into another project's git repository.

- `--with-rpath=<colon-separated-rpath>`

  Embed one or more library paths, so that libjemalloc can find the libraries it is linked to. This works only on ELF-based systems.

- `--with-mangling=<map>`

Mangle public symbols specified in which is a comma-separated list of name:mangled pairs.

For example, to use ld's --wrap option as an alternative method for overriding libc's malloc implementation, specify something like:

```
--with-mangling=malloc:__wrap_malloc,free:__wrap_free[...]
```

Note that mangling happens prior to application of the prefix specified by --with-jemalloc-prefix, and mangled symbols are then ignored when applying the prefix.

- `--with-jemalloc-prefix=<prefix>`

  Prefix all public APIs with . For example, if is "prefix_", API changes like the following occur:

  ```
  malloc()          --> prefix_malloc()
  malloc_conf       --> prefix_malloc_conf
  /etc/malloc.conf --> /etc/prefix_malloc.conf
  MALLOC_CONF       --> PREFIX_MALLOC_CONF
  ```

  This makes it possible to use jemalloc at the same time as the system allocator, or even to use multiple copies of jemalloc simultaneously.

  By default, the prefix is "", except on OS X, where it is "je_". On OS X, jemalloc overlays the default malloc zone, but makes no attempt to actually replace the "malloc", "calloc", etc. symbols.

- `--without-export`

  Don't export public APIs. This can be useful when building jemalloc as a static library, or to avoid exporting public APIs when using the zone allocator on OSX.

- `--with-private-namespace=<prefix>`

  Prefix all library-private APIs with je_. For shared libraries, symbol visibility mechanisms prevent these symbols from being exported, but for static libraries, naming collisions are a real possibility. By default, is empty, which results in a symbol prefix of je_ .

- `--with-install-suffix=<suffix>`

  Append to the base name of all installed files, such that multiple versions of jemalloc can coexist in the same installation directory. For example, libjemalloc.so.0 becomes libjemalloc.so.0.

- `--with-malloc-conf=<malloc_conf>`

  Embed `<malloc_conf>` as a run-time options string that is processed prior to the malloc_conf global variable, the /etc/malloc.conf symlink, and the MALLOC_CONF environment variable. For example, to change the default decay time to 30 seconds:

  ```
  --with-malloc-conf=decay_ms:30000
  ```

- `--enable-debug`

  Enable assertions and validation code. This incurs a substantial performance hit, but is very useful during application development.

- `--disable-stats`

Disable statistics gathering functionality. See the "opt.stats_print" option documentation for usage details.

- `--enable-prof`

  Enable heap profiling and leak detection functionality. See the "opt.prof" option documentation for usage details. When enabled, there are several approaches to backtracing, and the configure script chooses the first one in the following list that appears to function correctly:

  - libunwind (requires --enable-prof-libunwind)
  - libgcc (unless --disable-prof-libgcc)
  - gcc intrinsics (unless --disable-prof-gcc)

- `--enable-prof-libunwind`

  Use the libunwind library (http://www.nongnu.org/libunwind/) for stack backtracing.

- `--disable-prof-libgcc`

  Disable the use of libgcc's backtracing functionality.

- `--disable-prof-gcc`

  Disable the use of gcc intrinsics for backtracing.

- `--with-static-libunwind=<libunwind.a>`

  Statically link against the specified libunwind.a rather than dynamically linking with -lunwind.

- `--disable-fill`

  Disable support for junk/zero filling of memory. See the "opt.junk" and "opt.zero" option documentation for usage details.

- `--disable-zone-allocator`

  Disable zone allocator for Darwin. This means jemalloc won't be hooked as the default allocator on OSX/iOS.

- `--enable-utrace`

  Enable utrace(2)-based allocation tracing. This feature is not broadly portable (FreeBSD has it, but Linux and OS X do not).

- `--enable-xmalloc`

  Enable support for optional immediate termination due to out-of-memory errors, as is commonly implemented by "xmalloc" wrapper function for malloc. See the "opt.xmalloc" option documentation for usage details.

- `--enable-lazy-lock`

  Enable code that wraps pthread_create() to detect when an application switches from single-threaded to multi-threaded mode, so that it can avoid mutex locking/unlocking operations while in single-threaded mode. In practice, this feature usually has little impact on performance unless thread-specific caching is disabled.

- `--disable-cache-oblivious`

Disable cache-oblivious large allocation alignment for large allocation requests with no alignment constraints. If this feature is disabled, all large allocations are page-aligned as an implementation artifact, which can severely harm CPU cache utilization. However, the cache-oblivious layout comes at the cost of one extra page per large allocation, which in the most extreme case increases physical memory usage for the 16 KiB size class to 20 KiB.

- `--disable-syscall`

  Disable use of syscall(2) rather than {open,read,write,close}(2). This is intended as a workaround for systems that place security limitations on syscall(2).

- `--disable-cxx`

  Disable C++ integration. This will cause new and delete operator implementations to be omitted.

- `--with-xslroot=<path>`

  Specify where to find DocBook XSL stylesheets when building the documentation.

- `--with-lg-page=<lg-page>`

  Specify the base 2 log of the allocator page size, which must in turn be at least as large as the system page size. By default the configure script determines the host's page size and sets the allocator page size equal to the system page size, so this option need not be specified unless the system page size may change between configuration and execution, e.g. when cross compiling.

- `--with-lg-hugepage=<lg-hugepage>`

  Specify the base 2 log of the system huge page size. This option is useful when cross compiling, or when overriding the default for systems that do not explicitly support huge pages.

- `--with-lg-quantum=<lg-quantum>`

  Specify the base 2 log of the minimum allocation alignment. jemalloc needs to know the minimum alignment that meets the following C standard requirement (quoted from the April 12, 2011 draft of the C11 standard):

  > *The pointer returned if the allocation succeeds is suitably aligned so that it may be assigned to a pointer to any type of object with a fundamental alignment requirement and then used to access such an object or an array of such objects in the space allocated [...]*

  This setting is architecture-specific, and although jemalloc includes known safe values for the most commonly used modern architectures, there is a wrinkle related to GNU libc (glibc) that may impact your choice of . On most modern architectures, this mandates 16-byte alignment (=4), but the glibc developers chose not to meet this requirement for performance reasons. An old discussion can be found at https://sourceware.org/bugzilla/show_bug.cgi?id=206 . Unlike glibc, jemalloc does follow the C standard by default (caveat: jemalloc technically cheats for size classes smaller than the quantum), but the fact that Linux systems already work around this allocator noncompliance means that it is generally safe in practice to let jemalloc's minimum alignment follow glibc's lead. If you specify `--with-lg-quantum=3` during configuration, jemalloc will provide additional size classes that are not 16-byte-aligned (24, 40, and 56).

- `--with-lg-vaddr=<lg-vaddr>`

  Specify the number of significant virtual address bits. By default, the configure script attempts to detect virtual address size on those platforms where it knows how, and picks a default otherwise. This option may be useful when cross-compiling.

- `--disable-initial-exec-tls`

  Disable the initial-exec TLS model for jemalloc's internal thread-local storage (on those platforms that support explicit settings). This can allow jemalloc to be dynamically loaded after program startup (e.g. using dlopen). Note that in this case, there will be two malloc implementations operating in the same process, which will almost certainly result in confusing runtime crashes if pointers leak from one implementation to the other.

- `--disable-libdl`

  Disable the usage of libdl, namely dlsym(3) which is required by the lazy lock option. This can allow building static binaries.

The following environment variables (not a definitive list) impact configure's behavior:

- `CFLAGS="?"`
- `CXXFLAGS="?"`

  Pass these flags to the C/C++ compiler. Any flags set by the configure script are prepended, which means explicitly set flags generally take precedence. Take care when specifying flags such as -Werror, because configure tests may be affected in undesirable ways.

- `EXTRA_CFLAGS="?"`
- `EXTRA_CXXFLAGS="?"`

  Append these flags to CFLAGS/CXXFLAGS, without passing them to the compiler(s) during configuration. This makes it possible to add flags such as -Werror, while allowing the configure script to determine what other flags are appropriate for the specified configuration.

- `CPPFLAGS="?"`

  Pass these flags to the C preprocessor. Note that CFLAGS is not passed to 'cpp' when 'configure' is looking for include files, so you must use CPPFLAGS instead if you need to help 'configure' find header files.

- `LD_LIBRARY_PATH="?"`

  'ld' uses this colon-separated list to find libraries.

- `LDFLAGS="?"`

  Pass these flags when linking.

- `PATH="?"`

  'configure' uses this to find programs.

In some cases it may be necessary to work around configuration results that do not match reality. For example, Linux 4.5 added support for the MADV_FREE flag to madvise(2), which can cause problems if building on a host with MADV_FREE support and deploying to a target without. To work around this, use a cache file to override the relevant configuration variable defined in configure.ac, e.g.:

```
echo "je_cv_madv_free=no" > config.cache && ./configure -C
```

Advanced compilation

To build only parts of jemalloc, use the following targets:

```
build_lib_shared
build_lib_static
build_lib
build_doc_html
build_doc_man
build_doc
```

To install only parts of jemalloc, use the following targets:

```
install_bin
install_include
install_lib_shared
install_lib_static
install_lib_pc
install_lib
install_doc_html
install_doc_man
install_doc
```

To clean up build results to varying degrees, use the following make targets:

```
clean
distclean
relclean
```

Advanced installation

Optionally, define make variables when invoking make, including (not exclusively):

- `INCLUDEDIR="?"`

  Use this as the installation prefix for header files.

- `LIBDIR="?"`

  Use this as the installation prefix for libraries.

- `MANDIR="?"`

  Use this as the installation prefix for man pages.

- `DESTDIR="?"`

  Prepend DESTDIR to INCLUDEDIR, LIBDIR, DATADIR, and MANDIR. This is useful when installing to a different path than was specified via --prefix.

- `CC="?"`

  Use this to invoke the C compiler.

- `CFLAGS="?"`

  Pass these flags to the compiler.

- `CPPFLAGS="?"`

  Pass these flags to the C preprocessor.

- `LDFLAGS="?"`

  Pass these flags when linking.

- `PATH="?"`

  Use this to search for programs used during configuration and building.

Development

If you intend to make non-trivial changes to jemalloc, use the 'autogen.sh' script rather than 'configure'. This re-generates 'configure', enables configuration dependency rules, and enables re-generation of automatically generated source files.

The build system supports using an object directory separate from the source tree. For example, you can create an 'obj' directory, and from within that directory, issue configuration and build commands:

```
autoconf
mkdir obj
cd obj
../configure --enable-autogen
make
```

Documentation

The manual page is generated in both html and roff formats. Any web browser can be used to view the html manual. The roff manual page can be formatted prior to installation via the following command:

```
nroff -man -t doc/jemalloc.3
```