Octaspire Core Manual

www.octaspire.com

Table of Contents

Documentation for Octaspire Core library version 0.84.0

About

Octaspire Core is a container, utf-8 and utility library written in standard **C99**. It depends only on a C99 compiler and standard library.

Octaspire Core is used by some other Octaspire projects like Dern, Maze and Lightboard, but it should be usable on many different platforms and projects where containers like **vector**, **hash-map**, **pair**, **list**, **queue** or **utf-8 string** are needed.

Octaspire Core is licensed under the Apache License, Version 2.0. See file LICENSE in the source distribution for more information.

Octaspire Core uses Semantic Versioning 2.0.0 version numbering scheme. As long as the MAJOR version number is zero anything can change at any time, even in backwards incompatible manner.

NOTE	This library is not yet finished, and is still a work in progress to some extent. Unit tests and use in projects like Dern, Maze and Lightboard have helped to catch and fix quite a many bugs already, but there are still some features not yet implemented and probably bugs not yet fixed. Also, English in not my native language, so please bear with me, when reading this documentation.
NOTE	This documentation is work in progress.

Core uses Semantic Versioning 2.0.0 version numbering scheme. As long as the MAJOR version number is zero anything can change at any time, even in backwards incompatible manner.

Building the amalgamated source release

The amalgamated source release is the recommended way of using Core, if you don't need to modify Core itself. To use the amalgamated release, you will need only a C compiler and C standard library supporting C99.

Linux, FreeBSD, OpenBSD, NetBSD, OpenIndiana, DragonFly BSD, MidnightBSD, MINIX 3, Haiku, Syllable Desktop, macOS

```
curl -O octaspire.com/core/release.tar.bz2
tar jxf release.tar.bz2
cd release/*
curl -O https://octaspire.github.io/core/checksums
sha512sum -c checksums
sh how-to-build/YOUR_PLATFORM_NAME_HERE.XX
```

replace **YOUR_PLATFORM_NAME_HERE.XX** with **FreeBSD.sh**, **NetBSD.sh**, **OpenBSD.sh**, **OpenIndiana.sh**, **DragonFlyBSD.sh**, **MidnightBSD**, **linux.sh**, **minix3.sh**, **haiku.sh**, **SyllableDesktop.sh** or **macOS.sh**. More scripts for different platforms will be added later.

Windows using MinGW

1. Download and install **MinGW** from www.mingw.org into directory `C:\MinGW`. Please note, that you might need to add `C:\MinGW` and `C:\MinGW\bin` into the `PATH`. If you cannot install into `C:\MinGW`, you can install MinGW to some other place. Remember the installation path, because later you can write it into the `WindowsMinGW.bat` file, so that the script finds MinGW.
2. Download and install **7-Zip** from www.7-zip.org.
3. Download www.octaspire.com/core/release.tar.bz2 and extract it with 7-Zip. You might need to extract it twice; first into file `release.tar` and then again to get the directory.
4. Start **shell** and change directory to the extracted release directory and then into directory `version-x.y.z`, where x, y and z are some version numbers.
5. When you are in the directory `version-x.y.z` run command `how-to-build\WindowsMinGW.bat`

More scripts for different tools might be added later.

Running unit tests

This is all there should be to it; when make finishes, everything should be ready.

To run the unit tests:

```
test/octaspire-core-test-runner
```

NOTE	Man pages are not ready yet.

Quick tour of library features

Here we take a quick look at the features of the library.

Naming

C doesn't have namespaces. To prevent name collisions, all names in the library are prefixed with **octaspire_**. For example, **octaspire_container_utf8_string_t** is the string type and **octaspire_container_utf8_string_new** is one of the functions that create new strings. **octaspire_container_vector_t** is the vector type and **octaspire_container_hash_map_t** is the hash-map type.

These names can be pretty long, and quite annoying to write in the long run. I personally like to use the abbrevation features of vi/vim and emacs editors. You can see this setup in my dotfiles for nvi, vim and emacs editors. Your favorite editor or IDE might have similar functionality also available.

So, I can write **ocus** and hit the spacebar to get **octaspire_container_utf8_string_**. The table below lists some of the abbrevations I use:

ocus	octaspire_container_utf8_string_
ocve	octaspire_container_vector_
ochm	octaspire_container_hash_map_
ochme	octaspire_container_hash_map_element_

UTF-8 Strings

string-example.c

```
/*************************************************************************
 * To compile this file from the 'examples' directory:
 * c99 -Wall -Wextra -pedantic -I ../../include/ string-example.c
 *     -lm -L ../../build/ -loctaspire-core -o string-example
 *************************************************************************/
#include <stdio.h>
#define OCTASPIRE_CORE_AMALGAMATED_IMPLEMENTATION
#include "octaspire-core-amalgamated.c"

int main(void)
{
    octaspire_memory_allocator_t *allocator =
        octaspire_memory_allocator_new(0);

    octaspire_container_utf8_string_t *myStr =
        octaspire_container_utf8_string_new("Hello world!", allocator);

    printf(
        "String is \"%s\"\n",
        octaspire_container_utf8_string_get_c_string(myStr));

    octaspire_container_utf8_string_release(myStr);
    myStr = 0;

    octaspire_memory_allocator_release(allocator);
    allocator = 0;

    return 0;
}
```

Vector

vector-example.c

```
/*****************************************************************************
 * To compile this file from the 'examples' directory:
 * c99 -Wall -Wextra -pedantic -I ../../include/ vector-example.c
 *     -lm -L ../../build/ -loctaspire-core -o vector-example
 *****************************************************************************/
#include <stdio.h>
#define OCTASPIRE_CORE_AMALGAMATED_IMPLEMENTATION
#include "octaspire-core-amalgamated.c"

int main(void)
{
    octaspire_memory_allocator_t *allocator =
        octaspire_memory_allocator_new(0);

    octaspire_container_vector_t *v = octaspire_container_vector_new(
        sizeof(octaspire_container_utf8_string_t*),
        true,
        (octaspire_container_vector_element_callback_t)
            octaspire_container_utf8_string_release,
        allocator);

    for (size_t i = 0; i < 10; ++i)
    {
        octaspire_container_utf8_string_t *s =
            octaspire_container_utf8_string_new_format(
                allocator,
                "Hello %zu world!",
                i);

        if (!octaspire_container_vector_push_back_element(v, &s))
        {
            printf("Cannot insert string to vector\n");
        }
    }

    for (size_t i = 0; i < octaspire_container_vector_get_length(v); ++i)
    {
        octaspire_container_utf8_string_t const * const s =
            octaspire_container_vector_get_element_at_const(v, i);

        printf(
            "String %zu. is: %s\n",
            i,
            octaspire_container_utf8_string_get_c_string(s));
    }

    octaspire_container_vector_release(v);
```

```
    v = 0;

    octaspire_memory_allocator_release(allocator);
    allocator = 0;

    return 0;
}
```

List

TODO

Queue

TODO

Pair

TODO

Hash map

hash-map-example.c

```
/*****************************************************************
 * To compile this file from the 'examples' directory:
 * c99 -Wall -Wextra -pedantic -I ../../include/ hash-map-example.c
 *     -lm -L ../../build/ -loctaspire-core -o hash-map-example
 *****************************************************************/
#include <stdio.h>
#define OCTASPIRE_CORE_AMALGAMATED_IMPLEMENTATION
#include "octaspire-core-amalgamated.c"

int main(void)
{
    octaspire_memory_allocator_t *allocator =
        octaspire_memory_allocator_new(0);

    octaspire_container_hash_map_t * h =
        octaspire_container_hash_map_new_with_octaspire_container_utf8_string_keys(
            sizeof(float),
            false,
            (octaspire_container_hash_map_element_callback_function_t)0,
            allocator);


    for (size_t i = 0; i < 10; ++i)
    {
```

```
        octaspire_container_utf8_string_t *s =
            octaspire_container_utf8_string_new_format(allocator, "Item %zu!", i);

        float const value = i * 3.14;

        if (!octaspire_container_hash_map_put(
            h,
            octaspire_container_utf8_string_get_hash(s),
            &s,
            &value))
        {
            printf("Cannot insert element to hash map\n");
        }
    }

    for (size_t i = 0; i < octaspire_container_hash_map_get_number_of_elements(h);
++i)
    {
        octaspire_container_utf8_string_t *s =
            octaspire_container_utf8_string_new_format(allocator, "Item %zu!", i);

        octaspire_container_hash_map_element_t const * const element =
            octaspire_container_hash_map_get_const(
                h,
                octaspire_container_utf8_string_get_hash(s),
                &s);

        float const value =
            *(float const *
const)octaspire_container_hash_map_element_get_value(element);

        printf(
            "%s -> %f\n",
            octaspire_container_utf8_string_get_c_string(s),
            value);

        octaspire_container_utf8_string_release(s);
        s = 0;
    }

    octaspire_container_hash_map_release(h);
    h = 0;

    octaspire_memory_allocator_release(allocator);
    allocator = 0;

    return 0;
}
```

Input streams from memory and files

TODO

Lower level helpers

TODO

Building the development repository

Raspberry Pi, Debian and Ubuntu

To build Octaspire Core from source in Raspberry Pi, Debian or Ubuntu (16.04 LTS) system:

```
sudo apt-get install cmake git
git clone https://github.com/octaspire/core.git
cd core/build
cmake ..
make
```

Arch Linux

To build on Arch Linux (Arch Linux ARM) system:

```
sudo pacman -S cmake git gcc make
git clone https://github.com/octaspire/core.git
cd core/build
cmake ..
make
```

Haiku

To build on Haiku (Version Walter (Revision hrev51127) x86_gcc2):

```
pkgman install gcc_x86 cmake_x86
git clone https://github.com/octaspire/core.git
cd core/build
CC=gcc-x86 cmake ..
make
```

FreeBSD

To build on FreeBSD (FreeBSD-11.0-RELEASE-arm-armv6-RPI2) system:

```
sudo pkg install git cmake
git clone https://github.com/octaspire/core.git
cd core/build
cmake ..
make
```

MINIX 3

To build on MINIX 3 (minix_R3.3.0-588a35b) system:

```
su root
pkgin install cmake clang binutils git-base
exit
git clone git://github.com/octaspire/core
cd core/build
cmake ..
make
```

Other systems

On different systems the required commands can vary. In any case, You should install a **C compiler**, **cmake** and **git**.