ANSI C library API guide

Tom Kirchner **Version 1.0**
tom@tkirchner.com **January 8, 2014**

Contents

1 Introduction

1.1 About this document

This document specifies a generic format for ANSI C library APIs.

1.2 Intention of this guide

The intention behind this specification is the standardization of ANSI C library APIs so that the burden to use and integrate these kind of reuseable components is lowered.

There are a lot of C libraries in existance with different APIs and coding developed and in my opinion they would be much more easier to use if there was such a thing as a "standard C library interface definition". This document tries to define something like that.

2 Guidelines

A C API is defined inside a single `*.h` file that has the same name as the API itself.

This header file contains all public definitions of the API that a programmer needs to know in order to use it. The following "things"are allowed inside an API header file (usually in this order):

2.1 Type declarations

- All types must be defined using a typedef.
- All types must be opaque, meaning struct and union type members are not visible in the API.
- Define as few types as possible.
- Resort to as many native C data types as possible.
- Alias types are to be avoided.

2.2 API declaration

The API itself contains only functions. All API functions are contained in a typedef'ed struct that has the same name as the API itself.

2.3 Function declarations

There is exactly **one** public function defined in the API header file. It takes no arguments and is named as the API itself plus the suffix `GetContext`.

This function returns the static API struct with all its members initialized with the actual function implementations. No API function pointer is allowed to be 0. The API function names are not prefixed with the API name.

3 Example

This is an example API header file for an API named *myApi*. The file is named `myApi.h`:

```
typedef struct maTypeA maTypeA;
typedef struct maTypeB maTypeB;
typedef struct {
  void (*func_a)( void );
  void (*func_b)( void );
}
myApi;
extern myApi myApiGetContext( void );
```

4 Revision History

1.0 8 January 2014

Initial version