kLogger

Code documentation

01.09.2002

Beta 0.3.0

1. Introduction

This library is written in the C programming language for maximum performance. You can use it to easily and efficiently record data about the operation of a program.

2. API

2.1 __buffer* buffer_init(uint32_t bits)

First we initialize the data buffer with n bits, which will hold the data before it is written to disk. The size of the necessary buffer should be chosen wisely. The function returns a pointer to the buffer or zero if there is an error.

2.2 FILE* buffer_get_file(const char* str)

This function opens a thread to write data from the buffer to FILE and returns a pointer to it. If it does not find a file with that name, it creates one.

2.3 void buffer_reset(__buffer* buffer, FILE* file)

This function resets the entire buffer to the specified file.

2.4 void buffer_write(__buffer* buffer, const char* str, FILE* file)

This function writes the transmitted data to the specified buffer. To work correctly, you must specify the file in case there is not enough memory in the buffer.

2.5 void buffer_set(__buffer* buffer, uint32_t offset)

This function sets the size of the used memory to the "offset".

2.6 void buffer_free(__buffer* buffer)

This function frees previously allocated memory for the buffer.

3. Example programm

```
// Initialize the buffer with a size of 512 bits
    __buffer* buffer = buffer_init(512);

// Open a stream to write to a file
    FILE* file = buffer_get_file("logTest.txt");

// Writing different data to the buffer
    buffer_write(buffer, "HELLO WORLD", file);
    buffer_write(buffer, "\n1.1", file);
    buffer_write(buffer, "\n!@#$%^&*()_+", file);

// Dump the data into a file
    buffer_reset(buffer, file);
// Close the stream for recording.
    fclose(file);
// Free allocated memory
    buffer_free(buffer);
```