CS 180A Assignment 1a (Logger) - Ryman Barnett

Generated by Doxygen 1.8.17

1 File Index

File Index	1
1.1 File List	1
File Documentation	1
2.1 logger.c File Reference	1
2.1.1 Detailed Description	2
2.1.2 Function Documentation	3
2.2 logger.h File Reference	6
2.2.1 Detailed Description	7
2.2.2 Enumeration Type Documentation	7
2.2.3 Function Documentation	8
n <mark>dex</mark> 1	13

1 File Index

1.1 File List

Here is a list of all documented files with brief descriptions:

logger.c

Source file for logger functions to open and write to a file using buffered output that will dump to file when the buffer is full

logger.h

Header file for logger functions

2 File Documentation

2.1 logger.c File Reference

Source file for logger functions to open and write to a file using buffered output that will dump to file when the buffer is full.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <unistd.h>
#include "logger.h"
```

Functions

• LOG RESULT log open (const char *filename, unsigned int buffersize)

opens a file, and allocated a buffer for writting to it

• LOG RESULT log flush (void)

write contents of the buffer to the disk and then reset the buffer

LOG_RESULT log_write (const unsigned char *text)

write bytes from a given string into the buffer and flush it to disk when buffer is full

• LOG_RESULT log_close (void)

closes a file, deallocates all buffers, and resets counters

void log_dump (void)

dumps contents of the buffer to the command line

Variables

static unsigned char * LOG BUFFER

The memory buffer.

• static unsigned int LOG_BUFSIZE

The size of the buffer.

· static unsigned int LOG BYTECOUNT

How many bytes in the buffer?

• static char * LOG FILENAME

Name of the file to write the log to.

• static int LOG_FH = -1

The file handle of the open file.

2.1.1 Detailed Description

Source file for logger functions to open and write to a file using buffered output that will dump to file when the buffer is full.

Author

Ryman Barnett

email: ryman.b@digipen.edu

DigiPen login: ryman.b

Course: CS180

Section: A

Assignment #1a

Date

2022-09-13

2.1.2 Function Documentation

```
2.1.2.1 log_close() LOG_RESULT log_close ( void )
```

closes a file, deallocates all buffers, and resets counters

Returns

LOG WRITEFAIL if buffer could'nt be written to file, and LOG SUCCESS otherwise

Definition at line 167 of file logger.c.

```
168 {
        // flush remaining buffer into file
LOG_RESULT writeRes = log_flush();
169
170
171
        close(LOG_FH);  // close file
free(LOG_BUFFER);  // free buffer
free(LOG_FILENAME); // free filename
172
        close (LOG_FH);
173
174
175
         LOG_BUFFER = NULL; // reset dangling pointer
176
         LOG\_FH = -1;
                                        // reset dangling pointer
177
178 LOG_FILENAME = NULL;// reset dangling pointer
179 LOG_BUFSIZE = 0; // reset size counter
180 LOG_BYTECOUNT = 0; // reset bytes read
181
         return writeRes; // result of write
182
183 }
```

References LOG_BUFFER, LOG_BUFSIZE, LOG_BYTECOUNT, LOG_FH, LOG_FILENAME, and log_flush().

Referenced by log_flush(), and log_write().

```
2.1.2.2 log_flush() LOG_RESULT log_flush (
```

write contents of the buffer to the disk and then reset the buffer

Returns

LOG_WRITEFAIL if buffer cant be written to disk, and LOG_SUCCESS otherwise

Definition at line 86 of file logger.c.

```
88
     // try to write buffer to file
     if (LOG_BYTECOUNT == 0 || write(LOG_FH, LOG_BUFFER, LOG_BYTECOUNT))
89
90
       // sync file
91
       if (fsync(LOG_FH) != -1)
92
93
         LOG_BYTECOUNT = 0; // reset buffer counter
94
9.5
         return LOG_SUCCESS; // success
96
97
98
99
     log_close(); // close for error
100
101
     return LOG_WRITEFAIL; // failed to write or could not sync file
102
103 }
```

References LOG BUFFER, LOG BYTECOUNT, log close(), LOG FH, LOG SUCCESS, and LOG WRITEFAIL.

Referenced by log_close(), and log_write().

opens a file, and allocated a buffer for writting to it

Parameters

filename	const char* to the name of the file to open
buffersize	unsigned int to size of buffer to allocate

Returns

LOG_NOMEM if buffer cant be allocated, LOG_OPENFAIL if file not opened, and LOG_SUCCESS otherwise

Definition at line 44 of file logger.c.

```
45 {
     int len = (strlen(filename) + 1);
46
                                               // length of filename
    LOG BUFSIZE = buffersize;
                                               // store buffersize
47
    LOG_BYTECOUNT = 0;
                                               // Initial bytes read are 0
48
    LOG_FILENAME = (char *) malloc(len);
                                               // allocate space for filename
49
    LOG_BUFFER = (char *) malloc(LOG_BUFSIZE); // allocate buffer memory
50
51
    // make sure both mallocs were successfull
```

```
if (LOG_BUFFER == NULL || LOG_FILENAME == NULL)
55
      free(LOG_BUFFER); // free buffer if it was allocated
      free(LOG_FILENAME); // free filename if it was allocated
56
      return LOG_NOMEM; // memory not allocated
60
61
    // copy file name
    strcpy(LOG_FILENAME, filename);
62
65
    LOG_FH = open (LOG_FILENAME, O_WRONLY | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR);
    // validate opened file
    if (LOG_FH == -1)
69
      free(LOG_FILENAME); // free filename if it was allocated
70
                          // free buffer if it was allocated
      free (LOG_BUFFER);
72
      return LOG_OPENFAIL; // file not opened
73
74
75
76
    return LOG_SUCCESS; // no errors file was opened
```

References LOG_BUFFER, LOG_BUFSIZE, LOG_BYTECOUNT, LOG_FH, LOG_FILENAME, LOG_NOMEM, LOG← OPENFAIL, and LOG_SUCCESS.

write bytes from a given string into the buffer and flush it to disk when buffer is full

Parameters

text | const unsigned char* to text to write to file

Returns

LOG WRITEFAIL if buffer could'nt be written to file, and LOG SUCCESS otherwise

Definition at line 117 of file logger.c.

```
long unsigned i = 0; // loop counter
119
121
      // for each letter in text
122
      while (i < strlen(text))</pre>
124
        // check if it should flush
125
        if (LOG_BYTECOUNT < LOG_BUFSIZE)</pre>
126
127
         // if not write the text char into buffer
          LOG_BUFFER[LOG_BYTECOUNT] = text[i];
128
129
          ++LOG_BYTECOUNT; // another byte read
          ++i; // move text buffer counter
130
131
        else // if it should flush
132
133
          // flush buffer into file
134
          if (log_flush() == LOG_WRITEFAIL)
135
136
137
            log_close();
138
```

```
139
            return LOG_WRITEFAIL; // failed to write
140
141
142
143
144
     // check if final buffer is full
      if (LOG_BYTECOUNT >= LOG_BUFSIZE)
146
147
        // flush buffer into file
        if (log_flush() == LOG_WRITEFAIL)
148
150
         log_close();
151
          return LOG_WRITEFAIL; // failed to write
152
153
154 }
155
     return LOG_SUCCESS; // succeded
156
```

References LOG_BUFFER, LOG_BUFSIZE, LOG_BYTECOUNT, log_close(), log_flush(), LOG_SUCCESS, and LO← G WRITEFAIL.

2.2 logger.h File Reference

Header file for logger functions.

Typedefs

- typedef enum LOG_RESULT LOG_RESULT
 - < enum for return results of log functions

Enumerations

- enum LOG_RESULT { LOG_SUCCESS, LOG_NOMEM, LOG_OPENFAIL, LOG_WRITEFAIL }
 - < enum for return results of log functions

Functions

- LOG RESULT log open (const char *filename, unsigned int buffersize)
 - opens a file, and allocated a buffer for writting to it
- LOG_RESULT log_write (const unsigned char *text)

write bytes from a given string into the buffer and flush it to disk when buffer is full

• LOG RESULT log flush (void)

write contents of the buffer to the disk and then reset the buffer

LOG_RESULT log_close (void)

closes a file, deallocates all buffers, and resets counters

void log_dump (void)

dumps contents of the buffer to the command line

2.2.1 Detailed Description

Header file for logger functions.

Author

Ryman Barnett

email: ryman.b@digipen.edu

DigiPen login: ryman.b

Course: CS180

Section: A

Assignment #1a

Date

2022-09-13

2.2.2 Enumeration Type Documentation

2.2.2.1 LOG_RESULT enum LOG_RESULT

< enum for return results of log functions

Enumerator

LOG_SUCCESS	No errors.	
LOG_NOMEM	Couldn't allocate buffer.	
LOG_OPENFAIL	File couldn't be opened.	
	Disk write failed	
Ge ф@@<u>d</u>.₩/Р фТу <u>Б</u> фПА L	Blok Willo landa.	

Definition at line 18 of file logger.h.

```
19 {
20  LOG_SUCCESS,  //!< No errors
21  LOG_NOMEM,  //!< Couldn't allocate buffer
22  LOG_OPENFAIL,  //!< File couldn't be opened
23  LOG_WRITEFAIL,  //!< Disk write failed
24 }LOG_RESULT;</pre>
```

2.2.3 Function Documentation

```
2.2.3.1 log_close() LOG_RESULT log_close ( void )
```

closes a file, deallocates all buffers, and resets counters

Returns

LOG_WRITEFAIL if buffer could'nt be written to file, and LOG_SUCCESS otherwise

Definition at line 167 of file logger.c.

```
169
       // flush remaining buffer into file
170
      LOG_RESULT writeRes = log_flush();
171
172
      close(LOG_FH);
                              // close file
      free(LOG_BUFFER); // free buffer
free(LOG_FILENAME); // free filename
173
174
175
176
      LOG_BUFFER = NULL; // reset dangling pointer
177
      LOG\_FH = -1;
                              // reset dangling pointer
178
      LOG_FILENAME = NULL; // reset dangling pointer
      LOG_BUFSIZE = 0; // reset size counter
LOG_BYTECOUNT = 0; // reset bytes read
179
180
181
      return writeRes; // result of write
182
183 }
```

References LOG_BUFFER, LOG_BUFSIZE, LOG_BYTECOUNT, LOG_FH, LOG_FILENAME, and log_flush().

Referenced by log_flush(), and log_write().

```
2.2.3.2 log_flush() LOG_RESULT log_flush ( void )
```

write contents of the buffer to the disk and then reset the buffer

Returns

LOG_WRITEFAIL if buffer cant be written to disk, and LOG_SUCCESS otherwise

Definition at line 86 of file logger.c.

```
88
     // try to write buffer to file
89
    if (LOG_BYTECOUNT == 0 || write(LOG_FH, LOG_BUFFER, LOG_BYTECOUNT))
90
91
      // sync file
      if (fsync(LOG_FH) != -1)
92
93
      {
94
        LOG_BYTECOUNT = 0; // reset buffer counter
95
96
         return LOG_SUCCESS; // success
97
98
    }
99
100
    log_close(); // close for error
101
102
     return LOG_WRITEFAIL; // failed to write or could not sync file
103 }
```

References LOG_BUFFER, LOG_BYTECOUNT, log_close(), LOG_FH, LOG_SUCCESS, and LOG_WRITEFAIL.

Referenced by log_close(), and log_write().

opens a file, and allocated a buffer for writting to it

Parameters

filename	const char* to the name of the file to open
buffersize	unsigned int to size of buffer to allocate

Returns

LOG_NOMEM if buffer cant be allocated, LOG_OPENFAIL if file not opened, and LOG_SUCCESS otherwise

Definition at line 44 of file logger.c.

```
int len = (strlen(filename) + 1);
                                                   // length of filename
     LOG_BUFSIZE = buffersize;
                                                    // store buffersize
    LOG_BYTECOUNT = 0;
                                                   // Initial bytes read are 0
48
     LOG_FILENAME = (char *) malloc(len);
                                                    // allocate space for filename
50
    LOG_BUFFER = (char *) malloc(LOG_BUFSIZE); // allocate buffer memory
52
    // make sure both mallocs were successfull
     if (LOG_BUFFER == NULL || LOG_FILENAME == NULL)
53
54
      free(LOG_BUFFER); // free buffer if it was allocated
free(LOG_FILENAME); // free filename if it was allocated
55
56
       return LOG_NOMEM; // memory not allocated
58
59
    }
60
    // copy file name
61
```

```
strcpy(LOG_FILENAME, filename);

// open file

LOG_FH = open(LOG_FILENAME, O_WRONLY | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR);

// validate opened file

if (LOG_FH == -1)

free(LOG_FILENAME); // free filename if it was allocated

free(LOG_BUFFER); // free buffer if it was allocated

return LOG_OPENFAIL; // file not opened

return LOG_SUCCESS; // no errors file was opened

return LOG_SUCCESS; // no errors file was opened
```

References LOG_BUFFER, LOG_BUFSIZE, LOG_BYTECOUNT, LOG_FH, LOG_FILENAME, LOG_NOMEM, LOG← OPENFAIL, and LOG_SUCCESS.

write bytes from a given string into the buffer and flush it to disk when buffer is full

Parameters

text | const unsigned char* to text to write to file

Returns

LOG_WRITEFAIL if buffer could'nt be written to file, and LOG_SUCCESS otherwise

Definition at line 117 of file logger.c.

```
118 {
119
      long unsigned i = 0; // loop counter
120
121
      // for each letter in text
122
      while (i < strlen(text))</pre>
123
124
        // check if it should flush
125
        if (LOG_BYTECOUNT < LOG_BUFSIZE)</pre>
126
          // if not write the text char into buffer
          LOG_BUFFER[LOG_BYTECOUNT] = text[i];
128
129
          ++LOG_BYTECOUNT; // another byte read
130
          ++i; // move text buffer counter
131
132
        else // if it should flush
133
134
          // flush buffer into file
135
          if (log_flush() == LOG_WRITEFAIL)
136
137
            log_close();
138
139
            return LOG_WRITEFAIL; // failed to write
140
141
       }
142
      }
143
      // check if final buffer is full
144
145
      if (LOG_BYTECOUNT >= LOG_BUFSIZE)
146
        // flush buffer into file
147
```

Index

```
log_close
    logger.c, 3
    logger.h, 8
log_flush
    logger.c, 3
    logger.h, 8
LOG_NOMEM
    logger.h, 7
log_open
    logger.c, 4
    logger.h, 9
LOG_OPENFAIL
    logger.h, 7
LOG_RESULT
    logger.h, 7
LOG_SUCCESS
    logger.h, 7
log_write
    logger.c, 5
    logger.h, 10
LOG_WRITEFAIL
    logger.h, 7
logger.c, 1
    log_close, 3
    log_flush, 3
    log_open, 4
    log_write, 5
logger.h, 6
    log_close, 8
    log_flush, 8
    LOG_NOMEM, 7
    log open, 9
    LOG_OPENFAIL, 7
    LOG_RESULT, 7
    LOG_SUCCESS, 7
    log_write, 10
    LOG_WRITEFAIL, 7
```