Hleb

Generated by Doxygen 1.8.13

Contents

Chapter 1

Bug List

Member hleb_count_args (...)

This macro can count only up to 10. If param list is longer thar 10 elements behavior is undefined

Member hleb_log_var (var, func...)

If `func` returns poiter to allocated memory, that memory will leak

Chapter 2

File Index

2.1 File List

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

Chapter 3

File Documentation

3.1 attribute.h File Reference

This file defines useful attributes.

Macros

- #define **__error**(msg)
- #define **__warning**(msg)
- #define **__deprecated**(msg)
- #define **__pure**
- #define **__const**
- #define **__noreturn**
- #define **__noinline**
- #define **__inline**
- #define **__hot**
- #define **__cold**
- #define **__w_unused_res**
- #define **__constructor**(p)
- #define **__destructor**(p)
- #define **__cleanup**(f)

3.1.1 Detailed Description

This file defines useful attributes.

GCC has attributes that can be applied to functions or variables. While default syntax is ugly, there is defines for most useful of them.

3.2 escseq.h File Reference

This file defines most useful ANSI escape sequences.

Macros

- #define **CSI**
- #define **RESET**
- #define **BOLD**
- #define **ITALIC**
- #define **UNDERLINED**
- #define **BLINK**
- #define **CURUP**(n)
- #define **CURDOWN**(n)
- #define **CURFWD**(n)
- #define **CURBWD**(n)
- #define **CURNXTL**(n)
- #define **CURPRVL**(n)
- #define **CURPOS**(r, c)
- #define **CLRDISP**(p)
- #define **CLRLINE**(p)
- #define **SCRLUP**(n)
- #define **SCRLDWN**(n)
- #define **REPORT**
- #define **SAVEPOS**
- #define **RESTPOS**
- #define **SETCOLOR**(c)
- #define **RBLACK**
- #define **RRED**
- #define **RGREEN**
- #define **RYELLOW**
- #define **RBLUE**
- #define **RMAGNETA**
- #define **RCYAN**
- #define **RWHITE**
- #define **BBLACK**
- #define **BRED**
- #define **BGREEN**
- #define **BYELLOW**
- #define **BBLUE**
- #define **BMAGNETA**
- #define **BCYAN**
- #define **BWHITE**

3.2.1 Detailed Description

This file defines most useful ANSI escape sequences.

It's hard to remember tonns of constant values, so this file defines useful ANSI escape sequences

3.3 hleb.h File Reference

Library main file.

Macros

- #define hleb_stringize(e)

  Stringize argument.
- #define hleb_stringize_v(e)

  Stringize value of an argument.
- #define hleb_same_type(a, b)

  Check if two arguments is representatives of same type.
- #define hleb_typecheck(type, e)

  Compiletime type-checking assertion.
- #define hleb_must_be_array(a)

  Compiletime assertion that `a` is array.
- #define hleb_array_size(a)

  Get array capacity in compiletime.
- #define hleb_field_sizeof(s, f)

  Get sizeof of structs field without struct object.
- #define hleb_with_warn_ignored(warn, what...)

  Compile code ignoring specified warning.
- #define hleb_count_args(...)

  Get amount of variadic macro params.
- #define hleb_do_selected(pref, ...)

  Apply selected macro(or function) on args.
- #define hleb_first_arg(f, o...)

  Select first elem in comma-separated list.
- #define hleb_cut_first(f, o...)

  Cut first elem from comma-separated list.
- #define hleb_foreach(what, args...)

  Apply single-argument macro(or function) to each arg in list.
- #define hleb_ct_assert_msg(cond, msg)

  Compiletime assert with msg.
- #define hleb_ct_assert(cond)

  Compiletime assert.
- #define hleb_log_on(cond, msg)

  Print message to logfile if cond is false.
- #define hleb_warn_on(cond, msg)

  Similar to `hleb_log_on` except `msg` is yellow.
- #define hleb_err_on(cond, msg)

  Similar to `hleb_log_on` except `msg` is red.
- #define hleb_log_func

  Initiate function to be logged on start and end.
- #define hleb_log(msg)

  Unconditionally print message to log.
- #define hleb_log_var(var, func...)

  Dump var to log with its name and value.
- #define hleb_log_mvar(vars...)

  Dump multiple vars with one macro invocation.

3.3.1 Detailed Description

Library main file.

This file includes all auxiliary files and defines user-space macro names. This names, though starts with `hleb_` prefix, can be defined without prefix. Define `HLEB_SUPRESS_LIB_NAME` before inclusion to do so.

3.3.2 Macro Definition Documentation

3.3.2.1 hleb_array_size

```
#define hleb_array_size(
              a )
```

Get array capacity in compiletime.

Parameters

in	*a*	Array to count elements within

Returns

Amount of elements in array

Note

Will break compilation if `a` is not an array

3.3.2.2 hleb_count_args

```
#define hleb_count_args(
              ...  )
```

Get amount of variadic macro params.

Parameters

in	...	Param list

Returns

Amount of comma-separated params

Bug This macro can count only up to 10. If param list is longer thar 10 elements behavior is undefined

3.3.2.3 hleb_ct_assert

```
#define hleb_ct_assert(
              cond )
```

Compiletime assert.

Parameters

in	*cond*	Assert condition

Note

`cond` should be compiletime evaluated

3.3.2.4 hleb_ct_assert_msg

```
#define hleb_ct_assert_msg(
              cond,
              msg )
```

Compiletime assert with msg.

Parameters

in	*cond*	Assert condition
in	*msg*	Message to be shown is `cond` is false

Note

`cond` should be compiletime evaluated

3.3.2.5 hleb_cut_first

```
#define hleb_cut_first(
              f,
              o...  )
```

Cut first elem from comma-separated list.

Parameters

in	*Comma-separated*	list

Returns

List with first elem thrown away

3.3.2.6 hleb_do_selected

```
#define hleb_do_selected(
              pref,
              ... )
```

Apply selected macro(or function) on args.

Parameters

in	*pref*	Prefix of macro name
in	*...*	Argument list to be passed through

Note

This macro selects which macro to apply based on amount of arg list. Applied macro should have name `pref_N`, where N is param count.

3.3.2.7 hleb_field_sizeof

```
#define hleb_field_sizeof(
              s,
              f )
```

Get sizeof of structs field without struct object.

Parameters

in	*s*	struct name
in	*f*	field name

Returns

sizeof of structs field

3.3.2.8 hleb_first_arg

```
#define hleb_first_arg(
              f,
              o... )
```

Select first elem in comma-separated list.

Parameters

in	...	Comma-separated list

Returns

First list entry

3.3.2.9 hleb_foreach

```
#define hleb_foreach(
              what,
              args...  )
```

Apply single-argument macro(or function) to each arg in list.

Parameters

in	*what*	Single argument macro to apply
in	*args*	Comma-separated argument list

Note

This macro can be compiled if used with 10 or less args

3.3.2.10 hleb_log

```
#define hleb_log(
              msg )
```

Unconditionally print message to log.

Parameters

in	*msg*	Message to print

3.3.2.11 hleb_log_mvar

```
#define hleb_log_mvar(
              vars...  )
```

Dump multiple vars with one macro invocation.

Parameters

in	*vars*	Comma-separated list of variables

Note

This macro can't use optional `func`.

3.3.2.12 hleb_log_on

```
#define hleb_log_on(
              cond,
              msg )
```

Print message to logfile if cond is false.

Parameters

in	*cond*	Log contition
in	*msg*	Message to print

3.3.2.13 hleb_log_var

```
#define hleb_log_var(
              var,
              func...  )
```

Dump var to log with its name and value.

Parameters

in	*var*	Var to be dumped
in	*func*	Optional function to form dump of user-defined type

Note

`func` should get pointer to data and return `char[]` containing complete dump.

Bug If `func` returns poiter to allocated memory, that memory will leak

3.3.2.14 hleb_must_be_array

```
#define hleb_must_be_array(
              a )
```

Compiletime assertion that `a` is array.

Parameters

in	*a*	Expression to check

3.3.2.15 hleb_same_type

```
#define hleb_same_type(
              a,
              b )
```

Check if two arguments is representatives of same type.

Parameters

in	*a,b*	Expressions to type comparison

Returns

`1` in case params are representatives of same type, `0` otherwise

3.3.2.16 hleb_stringize

```
#define hleb_stringize(
              e )
```

Stringize argument.

Parameters

in	*e*	Value to stringize

Returns

Stringized param

3.3.2.17 hleb_stringize_v

```
#define hleb_stringize_v(
              e )
```

Stringize value of an argument.

Parameters

in	*e*	Value to stringize

Returns

Stringized param value

3.3.2.18 hleb_typecheck

```
#define hleb_typecheck(
              type,
              e )
```

Compiletime type-checking assertion.

Parameters

in	*type*	Type to check
in	*e*	Expression to check

3.3.2.19 hleb_with_warn_ignored

```
#define hleb_with_warn_ignored(
              warn,
              what...  )
```

Compile code ignoring specified warning.

Parameters

in	*warn*	Warning to ignore(example "-Wadress")
in	*what*	Code to compile

Note

While this macro can be useful in some circumstances, be careful. Ignoring warning is bad practice.

Index