zyerr

NAME

Error Dequeue Abstraction Library

LIBRARY

zyerr (-lzyerr -lzyalloc)

SYNOPSIS

```
#include <zyerr.h>

typedef struct zyerrbx_s zyerrbx_t;
typedef struct zyerr_s zyerr_t;

int zyerr_construct(zyerr_t **dqe, const zyalloc_t *alloc);
void zyerr_destruct(zyerr_t **dqe);
void zyerr_clear(zyerr_t *dqe);
int zyerr_push_first(zyerr_t *dqe, int64_t code, const char *file,
                     size_t line, const char *function,
                     const void *opaque, size_t opaque_size);
int zyerr_push_last(zyerr_t *dqe, int64_t code, const char *file,
                    size_t line, const char *function,
                    const void *opaque, size_t opaque_size);
void zyerr_discard_first(zyerr_t *dqe);
void zyerr_discard_last(zyerr_t *dqe);
zyerrbx_t *zyerr_peek_first(const zyerr_t *dqe);
zyerrbx_t *zyerr_peek_last(const zyerr_t *dqe);
size_t zyerr_size(const zyerr_t *dqe);
bool zyerr_is_empty(const zyerr_t *dqe);

int64_t zyerrbx_code(const zyerrbx_t *bx);
const char *zyerrbx_file(const zyerrbx_t *bx);
size_t zyerrbx_line(const zyerrbx_t *bx);
const char *zyerrbx_function(const zyerrbx_t *bx);
const void *zyerbx_opaque(const zyerrbx_t *bx, size_t *size);
```

DESCRIPTION

zyerr_construct()

`zyerr_construct` allocates a `zyerr_t` data structure using `alloc` and stores the result in `*dqe`. All function arguments must be non-null.

zyerr_destruct()

`zyerr_destruct` deallocates a `zyerr_t` data structure and sets `*dqe` to `nullptr`. Note that `dqe` must be non-null.

zyerr_clear()

`zyerr_clear` deallocates and unlinks all `zyerrbx_t` data structures stored in `dqe`. Note that `dqe` must be non-null.

zyerr_push_*()

`zyerr_push_first` and `zyerr_push_last` allocate a `zyerrbx_t` and store `code`, `file`, `line`, `function`,`opaque`, and `opaque_size`.

The resulting data structure is stored at the *front* and *back* of `dqe` for `zyerr_push_first` and `zyerr_push_last`, respectively.

Note that `dqe`, `file`, and `function` must be non-null and `line` must be non-zero. `opaque` may be set to `nullptr` and `opaque_size` to zero in order to indicate that there is no auxilliary data; however, if `opaque` is non-null, then `opaque_size` must be non-zero.

zyerr_discard_first()

`zyerr_discard_first` deallocates and unlinks the *front-most* `zyerrbx_t` data structure from `dqe`. Note that `dqe` must be non-null.

zyerr_discard_last()

`zyerr_discard_last` deallocates and unlinks the *back-most* `zyerrbx_t` data structure from `dqe`. Note that `dqe` must be non-null.

zyerr_peek_first()

`zyerr_peek_first` retrieves the *front-most* `zyerrbx_t` data structure from `dqe` if it exists. All function arguments must be non-null.

zyerr_peek_last()

`zyerr_peek_last` retrieves the *back-most* `zyerrbx_t` data structure from `dqe` if it exists. All function arguments must be non-null.

zyerr_size()

`zyerr_size` returns the number of elements stored in `dqe`. Note that `dqe` must be non-null.

zyerr_is_empty()

`zyerr_is_empty` returns a `true` if and only if there are no elements stored in `dqe`. Note that `dqe` must be non-null.

zyerrbx_code()

`zyerrbx_code` returns the *error code* associated with `bx`. Note that `bx` must be non-null.

zyerrbx_file()

`zyerrbx_file` returns the *file* associated with `bx`. Note that `bx` must be non-null.

zyerrbx_line()

`zyerrbx_line` returns the *line number* associated with `bx`. Note that `bx` must be non-null.

zyerrbx_function()

`zyerrbx_function` returns the *function name* associated with `bx`. Note that `bx` must be non-null.

zyerrbx_opaque()

`zyerrbx_opaque` returns the memory address of the opaque data associated with `bx` and stores its size in `*size`. Note that `bx` must be non-null. `size` may be `nullptr` to indicate that the size is known ahead of time and is unneeded.

RETURN VALUE

On success, `zyerr_construct`, `zyerr_push_first`, `zyerr_push_last` return `ZYERR_OK`. Otherwise, an error code is returned.

ERRORS

`zyerr_construct`, `zyerr_push_first`, and `zyerr_push_last` can fail with the following error.

ZYALLOC_ENOMEM Out of memory.

NOTES

It is undefined behavior to violate any preconditions of these functions (e.g. passing `nullptr` to a function argument that is specified as non-null).