zyalloc

NAME

Allocator Abstraction Library.

LIBRARY

zyalloc (-lzyalloc)

SYNOPSIS

```
#include <zyalloc.h>

typedef struct zyalloc_s zyalloc_t;
typedef void *(*zymalloc_t)(size_t size);
typedef void *(*zyrealloc_t)(void *ptr, size_t size);
typedef void (*zyfree_t)(void *ptr);

int zyalloc_construct(zyalloc_t **alloc, const zymalloc_t malloc,
                      const zyrealloc_t realloc, const zyfree_t free);
void zyalloc_destruct(zyalloc_t **alloc);
int zymalloc(const zyalloc_t *alloc, size_t size, void **ptr);
int zyrealloc(const zyalloc_t *alloc, size_t size, void **ptr);
void zyfree(const zyalloc_t *alloc, void **ptr);
```

DESCRIPTION

zyalloc_construct()

`zyalloc_construct` allocates a `zyalloc_t` data structure using `malloc` and stores the function pointers to `malloc`, `realloc`, and `free`.

All function arguments must be non-null. It is undefined behavior to input a nonconforming implementation of `malloc`, `realloc`, or `free`.

zyalloc_destruct()

`zyalloc_destruct` deallocates a `zyalloc_t` data structure and sets `*alloc` to `nullptr`.

Note that `alloc` must be non-null.

zymalloc()

`zymalloc` calls the `malloc` function pointer in `alloc` using `size` and assigns the result to `*ptr`.

All function arguments must be non-null.

zyrealloc()

`zyrealloc` calls the `realloc` function pointer in `alloc` using `size` and `*ptr` and assigns the result to `*ptr`. If the operation fails, `*ptr` is unchanged.

All function arguments must be non-null.

zyfree()

`zyfree` calls the `free` function pointer in `alloc` using `*ptr` and sets `*ptr` to `nullptr`.

All function arguments must be non-null.

RETURN VALUE

On success `zyalloc_construct`, `zymalloc`, and `zyrealloc` return `ZYALLOC_OK`. Otherwise an error code is returned.

ERRORS

`zyalloc_construct`, `zymalloc`, `zyrealloc` can fail with the following error.

ZYALLOC_ENOMEM Out of memory.

NOTES

The allocator must support the ISO C standard `malloc`, `realloc`, and `free` functions.

The allocator data structure is allocated and deallocated using the `malloc` and `free` function pointers.

It is undefined behavior to pass `NULL` to any of these functions.