# zy_alloc

## NAME

Allocator Abstraction Library.

## LIBRARY

zy_alloc (-lzy_alloc)

## SYNOPSIS

```
#include <zy_alloc.h>

typedef struct zy_alloc_s zy_alloc_t;
typedef void *(*zy_malloc_t)(size_t size);
typedef void *(*zy_realloc_t)(void *ptr, size_t size);
typedef void (*zy_free_t)(void *ptr);

int zy_alloc_construct(zy_alloc_t **alloc, const zy_malloc_t malloc,
                       const zy_realloc_t realloc, const zy_free_t free);
void zy_alloc_destruct(zy_alloc_t **alloc);
int zy_malloc(const zy_alloc_t *alloc, size_t size, void **ptr);
int zy_realloc(const zy_alloc_t *alloc, size_t size, void **ptr);
void zy_free(const zy_alloc_t *alloc, void **ptr);
```

## DESCRIPTION

### zy_alloc_construct()

`zy_alloc_construct` allocates a `zy_alloc_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`.

### zy_alloc_destruct()

`zy_alloc_destruct` deallocates a `zy_alloc_t` data structure and sets `*alloc` to `nullptr`.

Note that `alloc` must be non-null.

### zy_malloc()

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

All function arguments must be non-null.

### zy_realloc()

`zy_realloc` 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.

### zy_free()

`zy_free` 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 `zy_alloc_construct`, `zy_malloc`, and `zy_realloc` return `ZY_OK`; otherwise, an error code is returned.

## ERRORS

`zy_alloc_construct`, `zy_malloc`, `zy_realloc` can fail with the following error.

**ZY_E_NOMEM** 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.