offsetof

From Wikipedia, the free encyclopedia

C's **offsetof()** macro is an [ANSI C](http://en.wikipedia.org/wiki/ANSI_C) library feature found in [stddef.h](http://en.wikipedia.org/wiki/Stddef.h). It evaluates to the offset (in bytes) of a given member within a [struct](http://en.wikipedia.org/wiki/Struct) or[union](http://en.wikipedia.org/wiki/Union_(computer_science)) type, an expression of type [size\_t](http://en.wikipedia.org/wiki/Size_t). The offsetof() macro takes two [parameters](http://en.wikipedia.org/wiki/Parameters), the first being a structure name, and the second being the name of a member within the structure. It cannot be described as a C prototype.[[1]](http://en.wikipedia.org/wiki/Offsetof#cite_note-1)

[[edit](http://en.wikipedia.org/w/index.php?title=Offsetof&action=edit&section=1)]Implementation

The "traditional" implementation of the macro relied on the compiler being not especially picky about pointers; it obtained the offset of a member by specifying a hypothetical structure that begins at address zero:

#define offsetof(st, m) ((size\_t)(&((st \*)0)->m))

This works by casting a null pointer into a pointer to structure st, and then obtaining the address of member m within said structure.

While this works correctly in many compilers, it has [undefined behavior](http://en.wikipedia.org/wiki/Undefined_behavior) according to the C standard, since it involves a [dereference](http://en.wikipedia.org/wiki/Dereference) of a [null pointer](http://en.wikipedia.org/wiki/Null_pointer) (although, one might argue that no dereferencing takes place, because the whole expression is calculated at compile time). It also tends to produce confusing compiler diagnostics if one of the arguments is misspelled. Some modern compilers (such as [GCC](http://en.wikipedia.org/wiki/GNU_Compiler_Collection)) define the macro using a special form instead, e.g.[[2]](http://en.wikipedia.org/wiki/Offsetof#cite_note-2)

#define offsetof(st, m) \_\_builtin\_offsetof(st, m)

This builtin is especially useful with C++ classes or structs that declare a custom unary operator &.[[3]](http://en.wikipedia.org/wiki/Offsetof#cite_note-3)

[[edit](http://en.wikipedia.org/w/index.php?title=Offsetof&action=edit&section=2)]Usage

It is useful when implementing generic data structures in C. For example, the [Linux kernel](http://en.wikipedia.org/wiki/Linux_kernel) uses offsetof() to implement container\_of(), which allows something like a [mixin](http://en.wikipedia.org/wiki/Mixin) type to find the structure that contains it:[[4]](http://en.wikipedia.org/wiki/Offsetof#cite_note-4)

#define container\_of(ptr, type, member) ({ \

const typeof( ((type \*)0)->member ) \*\_\_mptr = (ptr); \

(type \*)( (char \*)\_\_mptr - offsetof(type,member) );})

This macro is used to retrieve an enclosing structure from a pointer to a nested element, such as this iteration of a linked list of my\_structobjects:

struct my\_struct {

const char \*name;

struct list\_node list;

};

**extern** struct list\_node \* list\_next(struct list\_node \*);

struct list\_node \*current = */\* ... \*/*

while(current != NULL){

struct my\_struct \*element = container\_of(current, struct my\_struct, list);

printf("%s**\n**", element->name);

current = list\_next(&element->list);

}