std::memcpy

Defined in header <cstring>

```
void* memcpy( void* dest, const void* src, std::size_t count );
```

Copies count bytes from the object pointed to by src to the object pointed to by dest. Both objects are reinterpreted as arrays of `unsigned char`.

If the objects overlap, the behavior is undefined.

If either dest or src is an invalid or null pointer, the behavior is undefined, even if count is zero.

If the objects are potentially-overlapping or not *TriviallyCopyable*, the behavior of memcpy is not specified and may be undefined (http://stackoverflow.com/questions/29777492) .

Parameters

dest	-	pointer to the memory location to copy to
src	-	pointer to the memory location to copy from
count	-	number of bytes to copy

Return value

dest

Notes

std::memcpy may be used to implicitly create objects in the destination buffer.

std::memcpy is meant to be the fastest library routine for memory-to-memory copy. It is usually more efficient than std::strcpy, which must scan the data it copies or std::memmove, which must take precautions to handle overlapping inputs.

Several C++ compilers transform suitable memory-copying loops to std::memcpy calls.

Where strict aliasing prohibits examining the same memory as values of two different types, std::memcpy may be used to convert the values.

Example

Run this code

```
#include <iostream>
#include <cstdint>
#include <cstring>

int main()
{
    // simple usage
    char source[] = "once upon a midnight dreary...", dest[4];
    std::memcpy(dest, source, sizeof dest);
    std::cout << "dest[4] = { ";
    for (char c : dest)
        std::cout << "'" << c << "', ";
    std::cout << "};\n";

    // reinterpreting
    double d = 0.1;
//  std::int64_t n = *reinterpret_cast<std::int64_t*>(&d); // aliasing violation
    std::int64_t n;
    std::memcpy(&n, &d, sizeof d); // OK

    std::cout << std::hexfloat << d << " is " << std::hex << n
              << " as an std::int64_t\n" << std::dec;

    // object creation in destination buffer
    struct S {
        int x{42};
        void print() const { std::cout << "{" << x << "}\n"; }
    } s;
    alignas(S) char buf[sizeof(S)];
    S* ps = new (buf) S; // placement new
    std::memcpy(ps, &s, sizeof s);
    ps->print();
}
```

Output:

```
dest[4] = { 'o', 'n', 'c', 'e', };
0x1.999999999999ap-4 is 3fb999999999999a as an std::int64_t
{42}
```

See also

memmove	moves one buffer to another (function)
memset	fills a buffer with a character (function)
wmemcpy	copies a certain amount of wide characters between two non-overlapping arrays (function)
copy	copies characters (public member function of std::basic_string<CharT,Traits,Allocator>)
copy **copy_if** (C++11)	copies a range of elements to a new location (function template)
copy_backward	copies a range of elements in backwards order (function template)
is_trivially_copyable (C++11)	checks if a type is trivially copyable (class template)
C documentation for **memcpy**	

Retrieved from "https://en.cppreference.com/mwiki/index.php?title=cpp/string/byte/memcpy&oldid=128010"