Subject: [solved]An U++ equivalent of bzero ? (if not a sin)
Posted by xrysf03 on Sat, 02 Nov 2019 18:34:42 GMT
View Forum Message <> Reply to Message

Dear gentlemen,

while messing with my toy proggie, I've reached a point where I need to initialize an array of "double" (the double-length floating point type) - as an accumulation buffer of sorts.

Defined roughly as

double my_accu_buf[SOME_PARTICULAR_INTEGER_SIZE];

Can I just use the bzero() function that I know from GNU Libc? The MinGW compiler behind U++ cannot find that function, even if I #include <strings.h> . Ahaa, memset() does work (I don't even need to #include <string.h>). It's true that "man bzero" says "nono, deprecated, use memset() instead". Or is there some U++ equivalent? Or, should I refrain from using the unsafe and ugly, plain old C arrays, and use some container template instead? Such as the Vector... And of course I can just iterate across the array, but that feels so *meh* :d

Come to think of that, if I zero-pad the storage allocated behind a "double", do I actually achieve the same as
double my_var = 0;  // ?

Recommendations welcome :)

Frank

Subject: Re: An U++ equivalent of bzero ? (if not a sin)
Posted by Oblivion on Sat, 02 Nov 2019 20:52:08 GMT
View Forum Message <> Reply to Message

Hello Frank,

And welcome to U++ forums.

Usually, the Upp::Zero() template works fine. :)

```
double darray[20];
Zero(darray);
```

Best regards,
Oblivion

Subject: Re: An U++ equivalent of bzero ? (if not a sin)
Posted by Klugier on Sat, 02 Nov 2019 21:26:26 GMT
View Forum Message <> Reply to Message

Hello Frank,

For plain arrays you can use std::fill from standard library:

```
#include <iostream>

int main() {
	double array[10];
	std::fill(std::begin(array), std::end(array), 0.0);

	for (int i = 0; i < 10; ++i) {
		std::cout << array[i] << "\n";
	}

	return 0;
}
```

Alternatively, you could use std::array that have compilation time defined size:

```
#include <iostream>
#include <array>

int main() {
	std::array<double, 10> array;
	array.fill(0.0);

	for (auto d : array) {
		std::cout << d << "\n";
	}

	return 0;
}
```

I prefer to use std::fill for plain arrays, because it is probable solution and works outside U++, however it requires c++17 standard. However, if your app is designed to work with Upp framework than you can freely use Upp::Zero, which is easy to use (easier than std::fill).

Sincerely,
Klugier

Subject: Re: An U++ equivalent of bzero ? (if not a sin)
Posted by xrysf03 on Sun, 03 Nov 2019 21:19:17 GMT
View Forum Message <> Reply to Message

...even better than I thought. Thanks for your exhaustive answers gentlemen :)

Frank

Subject: Re: An U++ equivalent of bzero ? (if not a sin)
Posted by xrysf03 on Sun, 03 Nov 2019 21:34:04 GMT
View Forum Message <> Reply to Message

Actually come to think of that... what if I allocate the buffer dynamically?

double* array_ptr = new double[some_calculated_size];

Will Upp::Zero() know the right size, by any chance? (Talk to the allocator behind the scenes? That would not be very good as a general approach, as in general a pointer needn't point to the beginning of an allocated chunk of memory, it can point to someplace inside an allocated buffer etc.) = I guess I'd better decide whether to use a proper container, or use manual iteration, or stick to memset()...

BTW that floating-point literal zero, written as 0.0, that's a nice trick I didn't know about, thanks Klugier :)

Subject: Re: An U++ equivalent of bzero ? (if not a sin)
Posted by Oblivion on Sun, 03 Nov 2019 22:08:36 GMT
View Forum Message <> Reply to Message

Hello Frank,

If you're working with U++ containers, but want to work with dynamic buffers with C-like

benefits: lets you avoid new/delete, for one. Also it let's you specifiy the initial value.):

```
Buffer<double> darray(200, 0.0);

for(int i = 0; i < 200; i++)
  Cout() << darray[i] << "\n";
```

Best regards,
Oblivion