Subject: Re: Adding GetCount/NumBytes to Buffer?
Posted by luoganda on Tue, 13 May 2025 11:23:30 GMT
View Forum Message <> Reply to Message

Updated:
Checked size of Vector<int>
and it's not that big compared to Buffer - i thought it's bigger, that's why proposal.
Down message is now there for info(eg also hint for UnifiedSizeOf in c/c++).
GetCount could still be added though(1stMessage).

First message version:
Quote:Since Buffer always has an info of NumItems allocated
i mostly meant that it has because it knows where it allocated it(so it can query size on that)
and/or also
when specifying either Buffer(size_t size) or Alloc(size_t size).
With that i mean - Buffer size can not be done with one of this two functions,
so because it has an unified alloc protocol - info is there and could be easily used for this,
since Buffer is smaller compared to Vector and more apropriate in some situations.
If Buffer is used elsewhere in func or down the code - it would be more apropriate
to have that NumBytes() to not make a messy code to pass extra numBytes to func or something else.

So, two solutions are here for adding GetCount:
1) without adding additional int num to Buffer and querying size dynamically(nonTrivial-butStillOk)
2) adding int num to Buffer and just set it whenever Buffer New is called

~~~~~~~
The second version of "Buffer always has an info of NumItems/NumBytes..."
i meant that it has an underlying info about it.

But then i reserched a little and found out that it has and it doesn't.
This should be there in c/c++(as an option since it could bloat compiler work - not end compiled code),
and would probably not be so hard to implement.

```
int anInt;
void *ptr=malloc(...)

//now, somewhere else in the code,
//how to check if it was allocated on a stack or on a heap?
//because when allocated on the stack one could simply do sizeof(anInt)
//and when on heap - one of heap size functions(in case of upp something else)
//I guess this could be a kind of Rtti info.
```
~~~~~~~

I meant if one had that info, NumBytes would be already there.
For now though - this is not so straightforward.

```
func(void*ptr){
  //unified sizeof
  bool stackBased=IsStackBased(ptr);
  bool heapBased=!stackBased;
  uint32 sz=UnifiedSizeOf(ptr);
  memset(ptr,0,sz);
}
```

I am not sure how this works,
when one allocates on the stack/heap - it probabaly has different segments of address space,
so it could be also known by that.