SCM Challenge: Unique Words

Honorable Mentions

Arthur O’Dwyer
2017-09-28

```
// Name: Tai Meng

#include <iostream>
using namespace std;

int main()
{
    // Max N = _652_ in this online editor
    cout << "1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77
[...]
\t82\t81\t80\t79\t78\t77\t76\t75\t74\t73\t72\t71\t70\t69\t68\t67\t66\t65\t64
\t63\t62\t61\t60\t59\t58\t57\t56\t55\t54\t53\t52\t51\t50\t49\t48\t47\t46\t45
\t44\t43\t42\t41\t40\t39\t38\t37\t36\t35\t34\t33\t32\t31\t30\t29\t28\t27\t26
\t25\t24\t23\t22\t21\t20\t19\t18\t17\t16\t15\t14\t13\t12\t11\t10\t9\t8\t7\t6
\t5\t4\t3\t2\t1";
}
```

```
// Name: Michael Kazakov

#include <iostream>

using namespace std;

__attribute__ ((optnone)) void operator "" _flipping_heck_06_30_am( decltype(0ULL) )
{
    for( ; stdin->_fileno + 100 - 99 < *((uint64_t*)__builtin_alloca(0L) + 9 - 6); )
        printf( "%u ", ((*(((unsigned*)_IO_stdin)+28))++) + 200 - 199 );
    while( ((cout << ((*((long*)alloca(0LL) + 3))--) << " "), true) )
        if( !*((int64_t*)__builtin_frame_address(0) - 1) )
            break;
}

int main()
{
    500_flipping_heck_06_30_am;
}
```

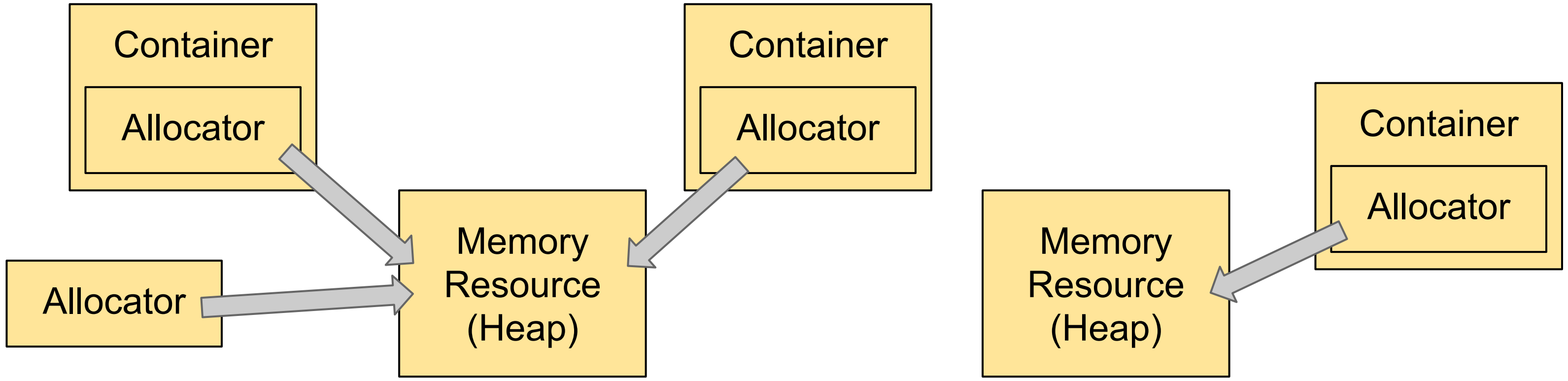
```
// Name: Richard Smith

unsigned n[] __asm__("main") __attribute__((section(".text"))) = {
0x41574155,0x41554156,0x01bf5354,0x49000000,0x000000be,0x00000f00,
0xffbf4100,0xbbffffff,0x0000000a,0x0000bd48,0xffff0000,0x8941ffff,
0xfd8941fc,0x44c644eb,0x3120f724,0xf6894cc9,0xebe08944,0xfbf79910,
0x8830c280,0x48f60c54,0xff48ee01,0x75c085c9,0xfec148ec,0xe6014820,
0xe8c68348,0xca29fa89,0x050ff889,0x8cfc8141,0x45000002,0x45ef440f,
0x8545ec01,0x31b775e4,0x5c415bc0,0x5e415d41,0xc35d5f41
};
```

The Allocator Model

An allocator is a handle to a heap

Clarifies our thinking about allocators

- Old-style thinking: “an allocator represents a source of memory” — ***WRONG!***
- New-style thinking: “an allocator represents *a pointer to* a source of memory (plus some orthogonal bits).”



Corollaries to the new way of thinking

- Allocator types should be copyable, just like pointers.
  - This was always true but now it’s more obvious.
- Allocator types should be **cheaply** copyable, like pointers.
  - They need not be **trivially** copyable.
  - Might it ever make sense to *reference-count* a heap?
- Memory-resource types should generally be immobile.
  - A memory resource might allocate chunks out of a buffer stored inside itself as a data member.

But what about stateless allocators?

A stateless allocator [e.g. `std::allocator<T>`] represents a pointer to a source of memory (plus some orthogonal bits) where that source of memory is a global singleton [e.g. the `new`/`delete` heap].

- A datatype with *k* possible values needs only $\log_2 k$ bits.
- A pointer to a global singleton (with 1 possible value) needs $\log_2 1 = 0$ bits.