STACK AND HEAP: COMMONLY ABUSED TERMS

Simon Brand
Codeplay Software Ltd.

AGENDA

- A bit about me
- What misuse am I talking about?
- Why is it wrong?
- What does the standard say?
- What terms should we use instead?

C++ AND ME

- Work with C++ daily
- Active on Stack Overflow (C++ gold badge)
- Technically on the standards committee
- Interested in metaprogramming and dark corners

WHAT MISUSE AM I TALKING ABOUT?

```
static int a;
static int b = 93;

void foo (int c) {
    int d = 42;
}

int main() {
    auto e = new int{314};
    foo(*e);
}
```

a	.bss binary section
b	.data binary section
c	register
d	stack
*e	heap

```
.file   "test.cpp"
        .intel_syntax noprefix
        .local  _ZL1a
        .comm   _ZL1a,4,4               ;a in .bss (name,size,align)
        .data                           ;b in .data
        .align 4
        .type   _ZL1b, @object
        .size   _ZL1b, 4
_ZL1b:
        .long   93
```

a	.bss binary section
b	.data binary section

```
main:
; ...
        call    _Znwm                   ;allocate e with new
        mov     DWORD PTR [rax], 314    ;store 314 at *e
        mov     QWORD PTR [rbp-8], rax  ;put e on stack
        mov     rax, QWORD PTR [rbp-8]
        mov     eax, DWORD PTR [rax]    ;put *e in register
        mov     edi, eax                ;put *e in arg register
        call    _Z3fooi
; ...
```

c	passed in register
*e	free store

```
_Z3fooi:                                ;start of foo
.LFB0:
        .cfi_startproc
        push    rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        mov     rbp, rsp
        .cfi_def_cfa_register 6
        mov     DWORD PTR [rbp-20], edi  ;move c from reg to stac
        mov     DWORD PTR [rbp-4], 42    ;d on stack
        nop
        pop     rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
```

c	passed in register, stored on stack
d	stack

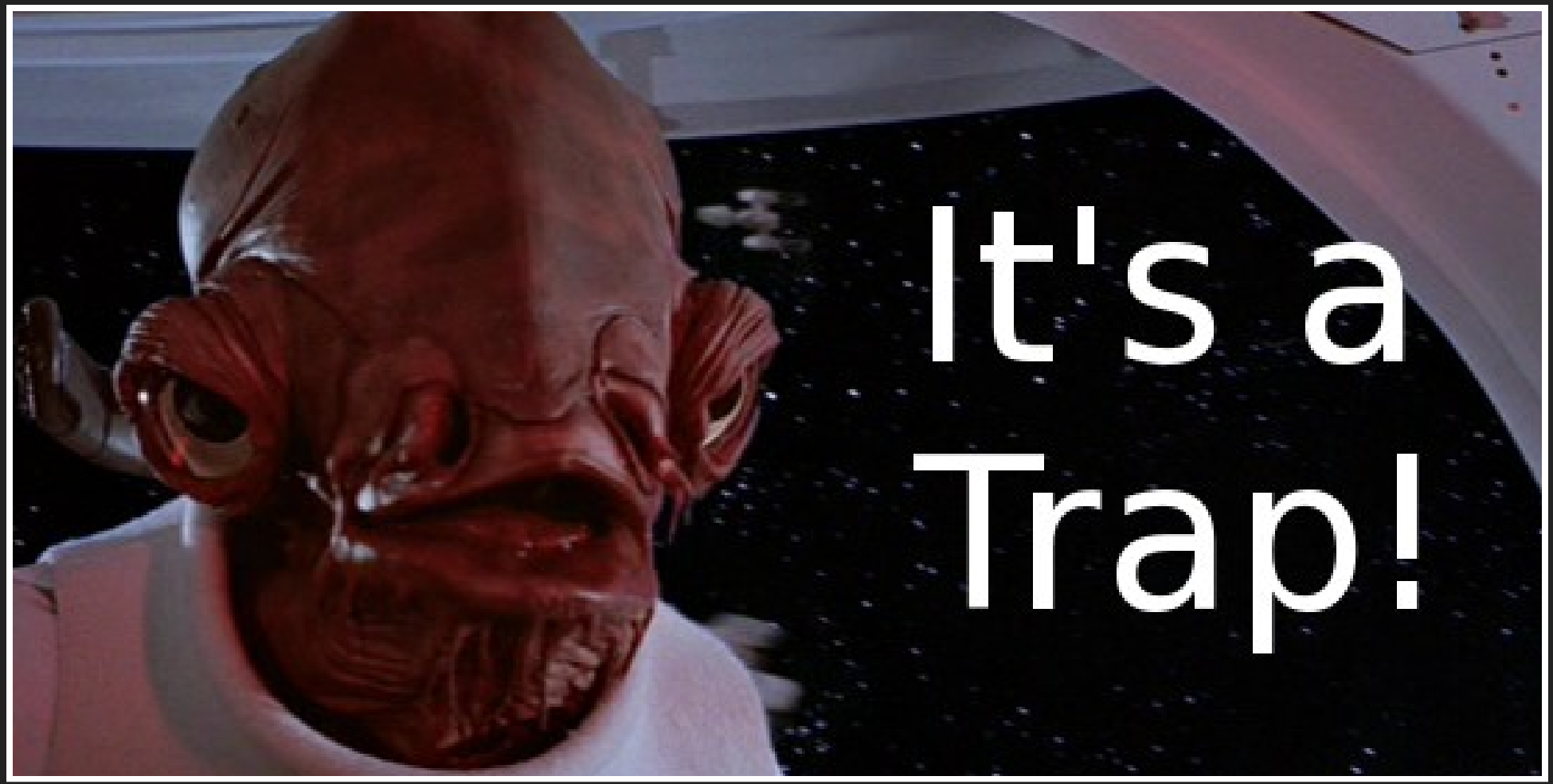
"CORRECT" ANSWER

```
static int a;
static int b = 93;

void foo (int c) {
    int d = 42;
}

int main() {
    auto e = new int{314};
    foo(*e);
}
```

a	.bss binary section
b	.data binary section
c	passed in register, stored on stack
d	stack
*e	free store


It's a
Trap!

WHY IS IT WRONG?

Lets turn on optimizations

```
static int a;
static int b = 4;
void foo (int c) {
    int d = 42;
}
int main() {
    auto e = new int{314};
    foo(*e);
}
```

a	Optimized out
b	Optimized out
c	Optimized out
d	Optimized out
*e	Free store

```
.file   "test.cpp"
        .intel_syntax noprefix
```

a	Optimized out
b	Optimized out

```
main:
.LFB1:
        .cfi_startproc
        sub     rsp, 8
        .cfi_def_cfa_offset 16
        mov     edi, 4
        call    _Znwm           ;allocates e
        xor     eax, eax
        add     rsp, 8
        .cfi_def_cfa_offset 8
        ret
        .cfi_endproc
```

c	Optimized out
*e	Free store

```
_Z3fooi:
.LFB0:
        .cfi_startproc
        rep ret
        .cfi_endproc
```

c	Optimized out
d	Optimized out

You can't know how things will be allocated in the general case.

What does the standard say about stacks and heaps?

NOTHING.

C++ is built on abstractions.

The standard does not define storage *location*, it defines storage *duration*.

[basic.stc]/1:

Storage duration is the property of an object that defines the minimum potential lifetime of the storage containing the object. The storage duration is determined by the construct used to create the object and is one of the following:

- static storage duration
- thread storage duration
- automatic storage duration
- dynamic storage duration

STATIC STORAGE DURATION

```
static int a;
static int b = 42;

void foo() {
    static int c = 4;
}

struct Bar {
    const static int d = 2;
};
```

THREAD STORAGE DURATION

```
thread_local int a;
thread_local int b = 42;

void foo() {
    thread_local int ill_formed;
    static thread_local int c;
}

struct Bar {
    thread_local int d;
};
```

AUTOMATIC STORAGE DURATION

```
void foo(int a) {
    int b;
    register int c;
}
```

DYNAMIC STORAGE DURATION

```
int* a = new int{};

void foo() {
    int* b = new int{};
}
```

What is the storage duration of the ints?

```
static int a;
static int b = 93;

void foo (int c) {
    int d = 42;
}

int main() {
    auto e = new int{314};
    foo(*e);
}
```

a	Static
b	Static
c	Automatic
d	Automatic
*e	Dynamic

A rule of thumb:

Only refer to the storage location if you need to discuss where a variable is physically located. In all other cases, refer to the storage duration

Blog: https://tartanllama.github.io

Email: simon@codeplay.com

Twitter: @TartanLlama

Codeplay: www.codeplay.com