Welcome

Marcel Ebmer
mebmer@ubimet.com

What I do

What I do

Tiago

- member states
- 16km grid
- <1 point in Aspen
- 137 levels
- supercomputer
- 1 hour
- Fortran

What I do

Tiago

- member states
- 16km grid
- <1 point in Aspen
- 137 levels
- supercomputer
- 1 hour
- Fortran

Marcel

→commercial customers

→90m grid

→500 points in Aspen

→surface only

→retail servers

→real time

→Matlab

fun with the with keyword

1. Motivation

fun with the with keyword

1. Motivation
2. MACRO_IMPLEMENTATION

I have a class...

```
class Light {
public:
    void turn_on();
    void turn_off();
    bool is_on() const;
};
```

...and an objective.

```
class Light {
public:
    void turn_on();
    void turn_off();
    bool is_on() const;
};


/*  objectives for this evening:
    1. turn light on
    2. prepare drinks
    3. turn light off
    4. watch a movie */
```

Give it a shot

```
void evening() {
    the_light.turn_on();
    prepare_drinks();
    the_light.turn_off();
    watch_movie();
}
```

Give it a shot

```
void evening() {
    the_light.turn_on();
    prepare_drinks();     // Out_of_beer exception thrown here
    the_light.turn_off();
    watch_movie();
}
```

Okay, no problem...

```
class Light_guard {
public:
    explicit Light_guard(Light& light) : l{&light} { l->turn_on(); }
    ~Light_guard() { l->turn_off(); }
    // ...appropriate move construction and assignment...
    // ...deleted copy construction and assignment...
private:
    Light* l;
};
```

Okay, no problem...

```
void evening() {
    Light_guard guard{the_light};
    prepare_drinks();
    watch_movie();
}
```

Okay, no problem...

```
void evening() {
    Light_guard guard{the_light};
    prepare_drinks();
    watch_movie();              // the_light still on
}
```

Yeah, forgot about that...

```
void evening() {
    {
        Light_guard{the_light};
        prepare_drinks();
    }
    watch_movie();
}
```

Yeah, forgot about that...

```
void evening() {
    {
        Light_guard{the_light}; // only a temporary
        prepare_drinks();
    }
    watch_movie();
}
```

Yeah, forgot about that...

```
void evening() {
    {
        Light_guard{the_light}; // only a temporary
        prepare_drinks();       // difficult to do in the dark
    }
    watch_movie();
}
```

Yeah, forgot about that...

```
void evening() {
    {
        Light_guard waldo{the_light}; // has a name now
        prepare_drinks();
    }
    watch_movie();
}
```

Yeah, forgot about that...

```
void evening() {
    {
        Light_guard waldo{the_light}; // has a name now
        // but compiler whines about unused variable
        prepare_drinks();
    }
    watch_movie();
}
```

Alright, this works!

```
void evening() {
    {
        Light_guard waldo{the_light};
        (void) waldo;
        prepare_drinks();
    }
    watch_movie();
}
```

But what I wanted was:

```
void evening() {
    with (Light_guard{the_light})
        prepare_drinks();
    watch_movie();
}
```

The with statement

```
void foo() {
    with (std::lock_guard<std::mutex>{the_mutex}) {
        something();
        other_thing();
    }
    something_else();
}
```

The with statement

```
void foo() {
    with (std::lock_guard<std::mutex>{the_mutex}) {
        something();
        other_thing();
    }
    something_else();
}


void bar() with (Pushed_matrix{}) {
    draw_stuff();
}
```

But there's no such thing

introducing BOOST_WITH

MACRO_IMPLEMENTATION

```
#define BOOST_WITH(what)
```

MACRO_IMPLEMENTATION

```
#define BOOST_WITH(what)                                    \
    if (auto boost_with_always_true                         \
            = boost::with_detail::make_true(what))
```

MACRO_IMPLEMENTATION

```cpp
// macro entry point, tag dispatch for static type checking
template <class T>
auto make_true(T&& what) {
    return make_true(forward<T>(what), is_move_constructible<T>{});
}
```

MACRO_IMPLEMENTATION

```
// macro entry point, tag dispatch for static type checking
template <class T>
auto make_true(T&& what) {
    return make_true(forward<T>(what), is_move_constructible<T>{});
}

// static error for non-movable types
template <class T>
auto make_true(T&&, false_type) {
    static_assert(is_move_constructible<T>::value,
                  "BOOST_WITH requires the scoped object's type to be“
                  " move constructible");
    return true_type{}; // never reached
}
```

MACRO_IMPLEMENTATION

```
// wraps an object of movable type
// and provides conversion to bool (always true)
template <class T>
struct always_true {
    explicit always_true(T what) : x{move(what)} {}
    constexpr operator bool() const { return true; }
    T x;
};
```

MACRO_IMPLEMENTATION

```
// wraps an object of movable type
// and provides conversion to bool (always true)
template <class T>
struct always_true {
    explicit always_true(T what) : x{move(what)} {}
    constexpr operator bool() const { return true; }
    T x;
};

// always_true<T> construction helper
template <class T>
auto make_true(T&& what, true_type) {
    static_assert(is_move_constructible<T>::value,
                  "this is a BOOST_WITH bug");
    return always_true<T>{forward<T>(what)};
}
```

Let's try this out

```
Light the_light;

int main() {
    BOOST_WITH(Light_guard{the_light}) {
        assert(the_light.is_on());
        prepare_drinks();
    }
    assert(!the_light.is_on());
    watch_movie();
}
```

Let's try this out

```
Light the_light;

int main() {
    BOOST_WITH(Light_guard{the_light}) {
        assert(the_light.is_on());
        prepare_drinks();
    }
    assert(!the_light.is_on());
    watch_movie();
}

// ...compiles...
```

Let's try this out

```
Light the_light;

int main() {
    BOOST_WITH(Light_guard{the_light}) {
        assert(the_light.is_on());
        prepare_drinks();
    }
    assert(!the_light.is_on());
    watch_movie();
}

// ...compiles...
// ...runs...
```

Let's try this out

```
Light the_light;

int main() {
    BOOST_WITH(Light_guard{the_light}) {
        assert(the_light.is_on());
        prepare_drinks();
    }
    assert(!the_light.is_on());
    watch_movie();
}

// ...compiles...
// ...runs...
// ...does not fail an assertion...
```

Let's try this out

```
Light the_light;

int main() {
    BOOST_WITH(Light_guard{the_light}) {
        assert(the_light.is_on());
        prepare_drinks();
    }
    assert(!the_light.is_on());
    watch_movie();
}

// ...compiles...
// ...runs...
// ...does not fail an assertion...

// ...unused variable 'boost_with_always_true'...
```

Now I’m under pressure

```
#define BOOST_WITH(what)
```

Now I'm under pressure

```
#define BOOST_WITH(what)                                    \
    for (auto boost_with_loop_once = make_pair(what, true); \
         boost_with_loop_once.second;                       \
         boost_with_loop_once.second = false)
```

Now I'm under pressure

```
#define BOOST_WITH(what)                                                    \
    static_assert(                                                          \
        is_move_constructible<decay_t<decltype(what)>>::value,              \
        "BOOST_WITH requires the scoped object's type to be "               \
        "move constructible");                                              \
    for (auto boost_with_loop_once = make_pair(what, true);                 \
         boost_with_loop_once.second;                                       \
         boost_with_loop_once.second = false)
```

Let's try again

```
Light the_light;

int main() {
    BOOST_WITH(Light_guard{the_light}) {
        assert(the_light.is_on());
        prepare_drinks();
    }
    assert(!the_light.is_on());
    watch_movie();
}
```

Let's try again

```
Light the_light;

int main() {
    BOOST_WITH(Light_guard{the_light}) {
        assert(the_light.is_on());
        prepare_drinks();
    }
    assert(!the_light.is_on());
    watch_movie();
}

// ...compiles...
```

Let's try again

```
Light the_light;

int main() {
    BOOST_WITH(Light_guard{the_light}) {
        assert(the_light.is_on());
        prepare_drinks();
    }
    assert(!the_light.is_on());
    watch_movie();
}

// ...compiles...
// ...runs...
```

Let's try again

```
Light the_light;

int main() {
    BOOST_WITH(Light_guard{the_light}) {
        assert(the_light.is_on());
        prepare_drinks();
    }
    assert(!the_light.is_on());
    watch_movie();
}

// ...compiles...
// ...runs...
// ...does not fail an assertion...
```

Let's try again

```
Light the_light;

int main() {
    BOOST_WITH(Light_guard{the_light}) {
        assert(the_light.is_on());
        prepare_drinks();
    }
    assert(!the_light.is_on());
    watch_movie();
}

// ...compiles...
// ...runs...
// ...does not fail an assertion...
// ...no compiler warnings...
```

Let's try again

```
g++ -Wall -std=c++14 -O2

movb   $0x1,0x2005f5(%rip)        # 600b80 <the_light>
callq  4006b0 <_Z14prepare_drinksv>
movb   $0x0,0x2005e9(%rip)        # 600b80 <the_light>
callq  4006c0 <_Z11watch_moviev>
```

Thank you!