

# {CTRACER CHEAT SHEET}

<https://graphics.rwth-aachen.de:9000/ptrettner/ctracer>

## GETTING STARTED

ctracer is a C++ library for high-performance profiling and benchmarking.

### CMake

```
add_subdirectory(extern/ctracer)
target_link_libraries(Target PUBLIC ctracer)
```

### Headers

```
// TRACE(), ct::cyclers, ct::current_cycles
#include <ctracer/trace.hh>
// ct::scope
#include <ctracer/scope.hh>
// config, IO
#include <ctracer/trace-config.hh>
```

### Usage

```
void foo()
{
    TRACE();
    // do something ...
}
```

Each `TRACE()` costs about 70–100 cycles in total.

## SCOPES

By default, all TRACES are collected into a global thread-local scope.

## BASICS

### Trace

```
// profiles until end of scope
TRACE();
// named trace (only literals allowed)
TRACE("custom name");
```

TRACES are RAII objects that profile a scope. They record the current CPU cycles at start and end and also on which core they were executed. The result is recorded into a `ct::scope` and can be analyzed later.

### Current Cycles

```
uint64_t c = ct::current_cycles();
```

Returns the current number of CPU cycles.

### Cycler

```
ct::cyclers c;
// do something ...
std::cout << c.elapsed_cycles() << std::endl;
```

The `ct::cyclers` is a small utility for measuring the number of elapsed cycles.