# Modern cpp 30 lectures — logging

Friday, December 25, 2020　　3:03 PM

Easylogging++

一共只有两个文件，一个是头文件，一个是普通C++源文件，用的时候直接放在项目里。

```
#include "easylogging++.h"
INITIALIZE_EASYLOGGINGPP   <- 展开后，定义了全局对象。

int main()
{
  LOG(INFO) << "My first info log";
}
```

↖ Info 级别的日志记录器，同时传递了文件名，行号，函数名等日志需要的信息。

编译:

```
g++ -std=c++17 test.cpp easylogging++.cc
```

性能跟踪。

可以在日志中记录程序执行的性能数据。用于性能跟踪的三个宏的用法：

```
#include <chrono>
#include <thread>
#include "easylogging++.h"
INITIALIZE_EASYLOGGINGPP

void foo()
{
  TIMED_FUNC(timer);
  LOG(WARNING) << "A warning message.";
}

void bar()
{
  using namespace std::literals;
  TIMED_SCOPE(timer1, "void bar()");
  foo();
  foo();
  TIMED_BLOCK(timer2, "a block") {
     foo();
     std::this_thread::sleep_for(100us);
  }
}

int main()
{
  el::Configurations conf{"log.conf"};
  el::Loggers::reconfigureAllLoggers(conf);
  bar();
}
```

- TIMED_FUNC 接受一个参数，是用于性能跟踪的对象的名字，能自动产生函数的名称。
- TIMED_SCOPE 接受两个参数，分别是用于性能跟踪的对象的名字，以及用来记录的名字。

输出:

```
I executed [void foo()] in [5 us].
```

记录崩溃日志。

通过定义宏 ELPP_FEATURE_CRASH_LOG，可启用崩溃日志。当程序崩溃时，会自动在日志中记录程序的调用栈信息。再利用 addr2line 这样的工具，就知道是哪一行发生了崩溃。

```
#include "easylogging++.h"
INITIALIZE_EASYLOGGINGPP

void boom()
{
  char* ptr = nullptr;
  *ptr = '\0';
}

int main()
{
  el::Configurations conf{"log.conf"};
  el::Loggers::reconfigureAllLoggers(conf);
  boom();
}
```

spdlog.

```
#include "spdlog/spdlog.h"

int main()
{
  spdlog::info("My first info log");
}
```

spdlog 不是使用 IO 流风格输出了，采用跟 python 里 str.format 一样：

```
spdlog::warn("Message with arg {}", 42);
spdlog::error("{0:d}, {0:x}, {0:o}, {0:b}", 42);
```

这就是 C++20 的 format 的风格。