University of Stuttgart
Cluster of Excellence in Simulation Technology

Easylogging++

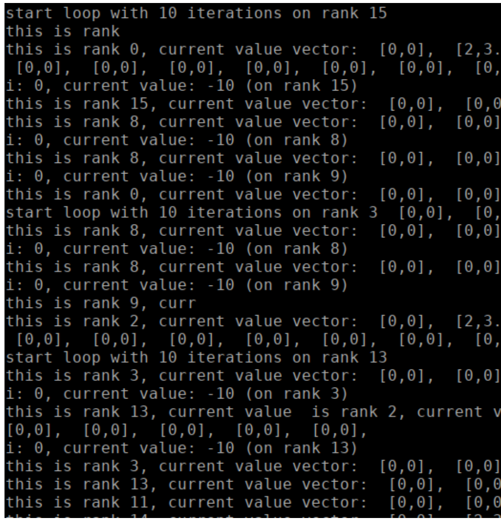
Group meeting 2019

Benjamin Maier

SimTech

Why would you need a logging library?

```cpp
const bool debug_output_1 = true;
const bool debug_output_2 = true;
const bool debug_output_3 = true;

std::vector<std::pair<int,float>> data(nIterations, std::pair<int,float>{});
int value = -10;

if (debug_output_1)
{
  int rankNo = 0;
  MPI_Comm_rank(MPI_COMM_WORLD, &rankNo);
  std::cout << "start loop with " << nIterations << " iterations on rank " << rankNo << std::endl;
}

for (int i = 0; i < nIterations; i++)
{
  // std::cout << "begin iteration " << i << std::endl;
  if (debug_output_2)
  {
    int rankNo = 0;
    MPI_Comm_rank(MPI_COMM_WORLD, &rankNo);
    std::cout << "this is rank " << rankNo << ", current value vector: ";
    for (std::vector<std::pair<int,float>>::const_iterator iter = data.begin(); iter != data.end(); iter++)
    {
      std::cout << " [" << iter->first << "," << iter->second << "], ";
    }
    std::cout << std::endl;
  }

  // std::cout << "now call computeData, in line 40 of code" << std::endl;
  computeData(2*i, data[i]);

  //std::cout << data[j].first << "," << data[j].second << std::endl;

  if (data[i].first < 0)
  {
    std::cout << "FATAL ERROR! This should not happen. In line 49 of file!";
    MPI_Abort(MPI_COMM_WORLD, 0);
  }

  value += i;

  if (debug_output_1)
  {
    if (i % 1000 == 0)
    {
      int rankNo = 0;
      MPI_Comm_rank(MPI_COMM_WORLD, &rankNo);
      std::cout << "i: " << i << ", current value: " << value << " (on rank " << rankNo << ")" << std::endl;
    }
  }
}
```



```
start loop with 10 iterations on rank 15
this is rank
this is rank 0, current value vector:  [0,0],  [2,3.
 [0,0],  [0,0],  [0,0],  [0,0],  [0,0],  [0,0],  [0,
i: 0, current value: -10 (on rank 15)
this is rank 15, current value vector:  [0,0],  [0,0
this is rank 8, current value vector:  [0,0],  [0,0]
i: 0, current value: -10 (on rank 8)
this is rank 8, current value vector:  [0,0],  [0,0]
i: 0, current value: -10 (on rank 9)
this is rank 0, current value vector:  [0,0],  [0,0]
start loop with 10 iterations on rank 3  [0,0],  [0,
this is rank 8, current value vector:  [0,0],  [0,0]
i: 0, current value: -10 (on rank 8)
this is rank 8, current value vector:  [0,0],  [0,0]
i: 0, current value: -10 (on rank 9)
this is rank 9, curr
this is rank 2, current value vector:  [0,0],  [2,3.
 [0,0],  [0,0],  [0,0],  [0,0],  [0,0],  [0,0],  [0,
start loop with 10 iterations on rank 13
this is rank 3, current value vector:  [0,0],  [0,0]
i: 0, current value: -10 (on rank 3)
this is rank 13, current value  is rank 2, current v
[0,0],  [0,0],  [0,0],  [0,0],  [0,0],
i: 0, current value: -10 (on rank 13)
this is rank 3, current value vector:  [0,0],  [0,0]
this is rank 13, current value vector:  [0,0],  [0,0
this is rank 11, current value vector:  [0,0],  [0,0
```

Why would you need a logging library?

Verbosity switches

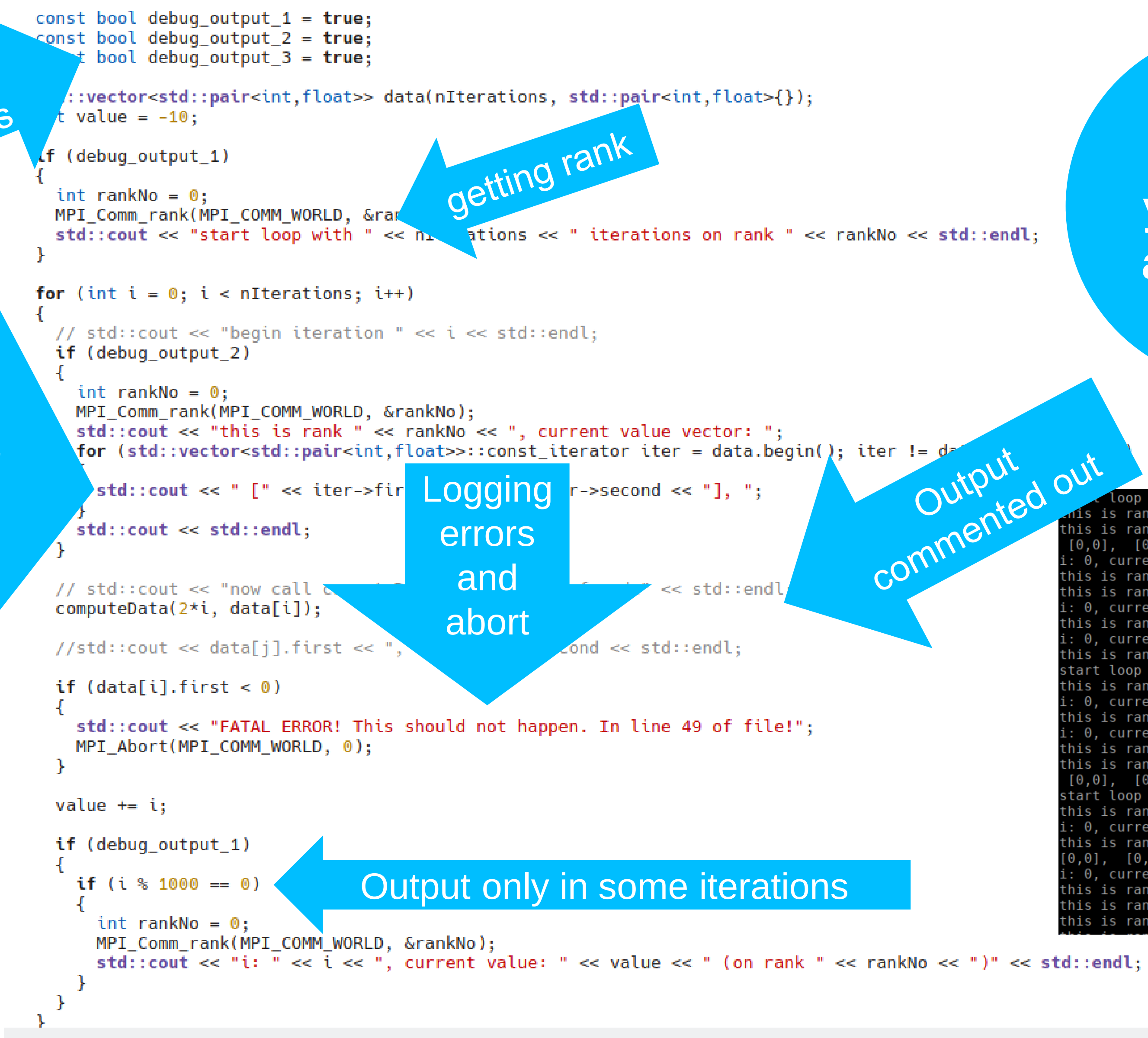
getting rank

Manually traversing data structure

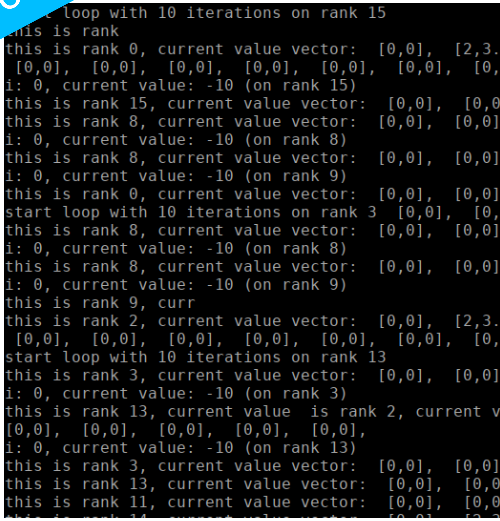
Logging errors and abort

Output commented out

Output only in some iterations



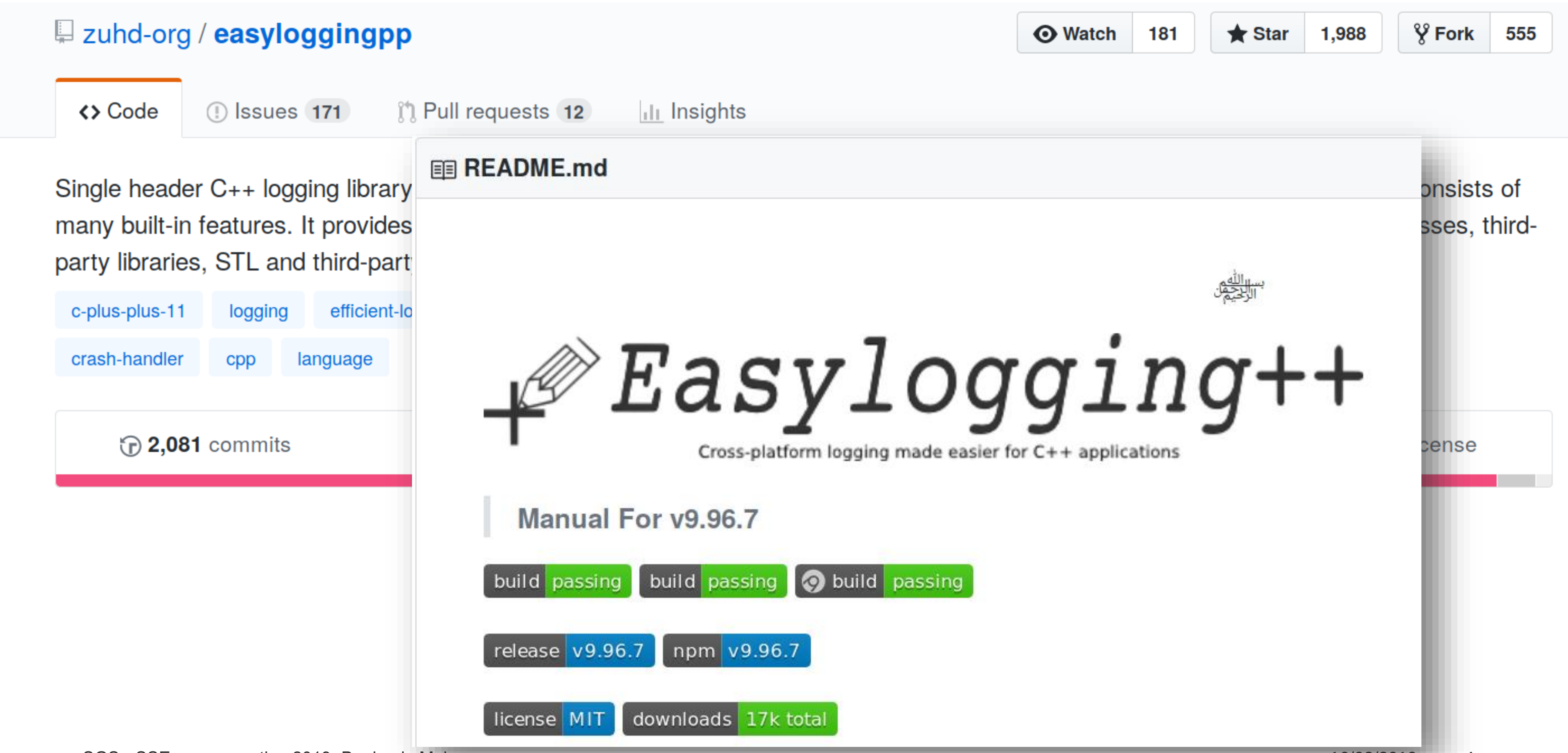
```
const bool debug_output_1 = true;
const bool debug_output_2 = true;
    t bool debug_output_3 = true;

   ::vector<std::pair<int,float>> data(nIterations, std::pair<int,float>{});
   t value = -10;

  if (debug_output_1)
{
  int rankNo = 0;
  MPI_Comm_rank(MPI_COMM_WORLD, &ran
  std::cout << "start loop with " << n     ations << " iterations on rank " << rankNo << std::endl;
}

for (int i = 0; i < nIterations; i++)
{
  // std::cout << "begin iteration " << i << std::endl;
  if (debug_output_2)
  {
    int rankNo = 0;
    MPI_Comm_rank(MPI_COMM_WORLD, &rankNo);
    std::cout << "this is rank " << rankNo << ", current value vector: ";
    for (std::vector<std::pair<int,float>>::const_iterator iter = data.begin(); iter != da
    
      std::cout << " [" << iter->fir                  r->second << "], ";
    }
    std::cout << std::endl;
  }

  // std::cout << "now call c                              " << std::endl
  computeData(2*i, data[i]);

  //std::cout << data[j].first << ",                    cond << std::endl;

  if (data[i].first < 0)
  {
    std::cout << "FATAL ERROR! This should not happen. In line 49 of file!";
    MPI_Abort(MPI_COMM_WORLD, 0);
  }

  value += i;

  if (debug_output_1)
  {
    if (i % 1000 == 0)
    {
      int rankNo = 0;
      MPI_Comm_rank(MPI_COMM_WORLD, &rankNo);
      std::cout << "i: " << i << ", current value: " << value << " (on rank " << rankNo << ")" << std::endl;
    }
  }
}
```



```
    loop with 10 iterations on rank 15
this is rank
this is rank 0, current value vector:  [0,0],  [2,3.
 [0,0],  [0,0],  [0,0],  [0,0],  [0,0],  [0,0],  [0,
i: 0, current value: -10 (on rank 15)
this is rank 15, current value vector:  [0,0],  [0,0
this is rank 8, current value vector:  [0,0],  [0,0]
i: 0, current value: -10 (on rank 8)
this is rank 8, current value vector:  [0,0],  [0,0]
i: 0, current value: -10 (on rank 9)
this is rank 0, current value vector:  [0,0],  [0,0]
start loop with 10 iterations on rank 3  [0,0],  [0,
this is rank 8, current value vector:  [0,0],  [0,0]
i: 0, current value: -10 (on rank 8)
this is rank 8, current value vector:  [0,0],  [0,0]
i: 0, current value: -10 (on rank 9)
this is rank 9, curr
this is rank 2, current value vector:  [0,0],  [2,3.
 [0,0],  [0,0],  [0,0],  [0,0],  [0,0],  [0,0],  [0,
start loop with 10 iterations on rank 13
this is rank 3, current value vector:  [0,0],  [0,0]
i: 0, current value: -10 (on rank 3)
this is rank 13, current value  is rank 2, current v
[0,0],  [0,0],  [0,0],  [0,0],  [0,0],
i: 0, current value: -10 (on rank 13)
this is rank 3, current value vector:  [0,0],  [0,0]
this is rank 13, current value vector:  [0,0],  [0,0
this is rank 11, current value vector:  [0,0],  [0,0
```

Overview

What is Easylogging++?



zuhd-org / easyloggingpp

Watch 181 Star 1,988 Fork 555

Code Issues 171 Pull requests 12 Insights

Single header C++ logging library … onsists of many built-in features. It provides … sses, third-party libraries, STL and third-part…

c-plus-plus-11 logging efficient-lo… crash-handler cpp language

2,081 commits … cense

README.md

Easylogging++

Cross-platform logging made easier for C++ applications

Manual For v9.96.7

build passing build passing build passing

release v9.96.7 npm v9.96.7

license MIT downloads 17k total

Overview

What is Easylogging++?

- easylogging++.h, easylogging++.cc
- No dependencies
- Logging to console and file, separate files for MPI ranks
- Different severity levels, e.g. Debug, Info, Error
- Different verbosity levels
- Logging disabled when building in release mode

→ demo0, demo1

Configuration

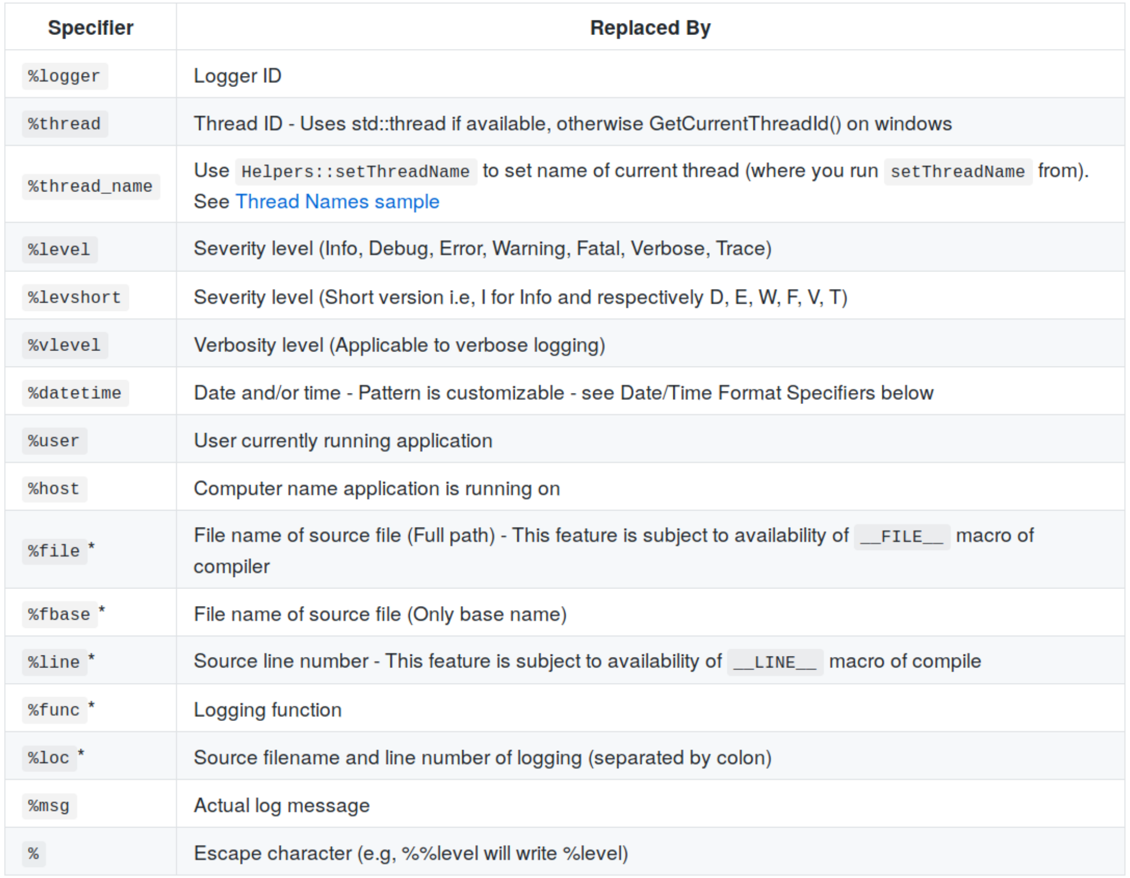
- Different log levels:

VLOG(1)	Verbose logging with verbosity level, only enabled when debugging the specific part of the code
LOG(DEBUG)	Debugging output, very frequently
LOG(TRACE)	Log where we are in the code
LOG(INFO)	Normal information message, only occasionally
LOG(WARNING)	Warning: user might have done something wrong
LOG(ERROR)	Error: something is wrong, program tries to continue
LOG(FATAL)	Fatal error, program aborts

- Behaviour can be configured
  - In a file → show file
  - Through the API in the code

Configuration

- Configuration for each level



Specifier	Replaced By
`%logger`	Logger ID
`%thread`	Thread ID - Uses std::thread if available, otherwise GetCurrentThreadId() on windows
`%thread_name`	Use `Helpers::setThreadName` to set name of current thread (where you run `setThreadName` from). See Thread Names sample
`%level`	Severity level (Info, Debug, Error, Warning, Fatal, Verbose, Trace)
`%levshort`	Severity level (Short version i.e, I for Info and respectively D, E, W, F, V, T)
`%vlevel`	Verbosity level (Applicable to verbose logging)
`%datetime`	Date and/or time - Pattern is customizable - see Date/Time Format Specifiers below
`%user`	User currently running application
`%host`	Computer name application is running on
`%file` *	File name of source file (Full path) - This feature is subject to availability of `__FILE__` macro of compiler
`%fbase` *	File name of source file (Only base name)
`%line` *	Source line number - This feature is subject to availability of `__LINE__` macro of compile
`%func` *	Logging function
`%loc` *	Source filename and line number of logging (separated by colon)
`%msg`	Actual log message
`%`	Escape character (e.g, %%level will write %level)

→ demo2, demo3

Verbose logging

- VLOG(1) to VLOG(9)
- Not shown by default, can be enabled on the command line
- Useful for inner loops or output of lots of data
-

`-v`	show all VLOGs
`--v=2`	show all VLOGs up to level 2
-vmodule=file.cpp=2,other_file.cpp=1	enable logs to level 2 in file.cpp and level 1 in other_file.cpp
-vmodule=function_*=5	enable verbosity level 5 in all files starting with function_

→ demo4

Additional features

- `if (VLOG_IS_ON(2)) { … }`
- Occasional logging:
  - `LOG_EVERY_N(1e3, DEBUG) << …`
  - `LOG_AFTER_N(10, ERROR) << “10 errors occured”;`
  - `LOG_N_TIMES(10, INFO) << “object initialized“;`
  - `LOG_IF(i > 5, DEBUG) << “Iteration “ << i;`
- Different custom loggers, e.g. for numerics, I/O, coupling etc.
  - Create different log files → demo5
- CHECK macros, like C++ `assert()`; produce FATAL error.
  - `CHECK(nNodes > 0) << “Number of nodes is invalid";`
    `CHECK_BOUNDS(i, 10, 20) << “Index is out of bounds”;`
    `CHECK_EQ, CHECK_LT, CHECK_LE, …`

Verbose logging

- Does it cost performance in release mode? → Demo6

```
Executed [debug outputs] in [861 ms]
Executed [log every n] in [844 ms]
Executed [verbose outputs] in [843 ms]
Executed [expensive function] in [6 seconds]
Executed [expensive function wrapped in VLOG_IS_ON] in [2 seconds]
```

Logging commented out:

```
Executed [debug outputs] in [285 ms]                            Overhead: (861-285)/1e8 = 5.7 ns per call
Executed [log every n] in [263 ms]                              Overhead: 5.8 ns per call
Executed [verbose outputs] in [264 ms]                          Overhead: 5.8 ns per call
Executed [expensive function] in [0 ms]                         Overhead: here: 600 ms per function call
Executed [expensive function wrapped in VLOG_IS_ON] in [259 ms] Overhead: 2.6 µs per call
```

Output STL containers

- To log other objects than fundamental types, overload the << operator:

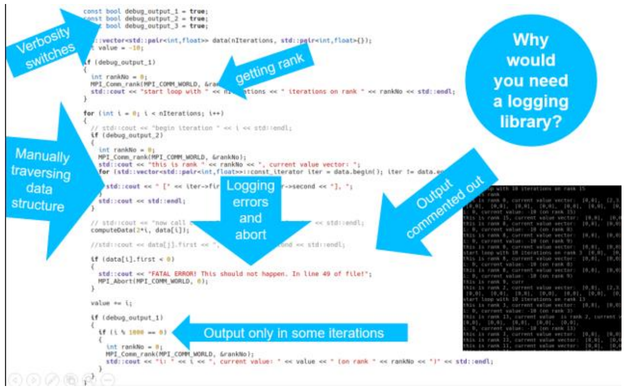
```
struct OwnClass
{
  int a;
  int b;
};

std::ostream &operator<<(std::ostream &stream, const OwnClass &rhs)
{
  stream << "{OwnClass a: " << rhs.a << ", b: " << rhs.b << "}";
  return stream;
}
```

→ Demo7

Conclusion

- Easylogging++ provides a convenient way to organize logging, especially when using MPI
- Debugging output can be kept in the code with VLOG after testing is done
- Control of output without recompilation
- Performance impact has to be considered

- Now your task is to fix the motivation problem: → demox



```
const bool debug_output_1 = true;
const bool debug_output_2 = true;
  t bool debug_output_3 = true;

 ::vector<std::pair<int,float>> data(nIterations, std::pair<int,float>{});
  t value = -10;

 if (debug_output_1)
{
  int rankNo = 0;
  MPI_Comm_rank(MPI_COMM_WORLD, &ra
  std::cout << "start loop with " << ni    ations << " iterations on rank " << rankNo << std::endl;
}

for (int i = 0; i < nIterations; i++)
{
  // std::cout << "begin iteration " << i << std::endl;
  if (debug_output_2)
  {
    int rankNo = 0;
    MPI_Comm_rank(MPI_COMM_WORLD, &rankNo);
    std::cout << "this is rank " << rankNo << ", current value vector: ";
    for (std::vector<std::pair<int,float>>::const_iterator iter = data.begin(); iter != data.e
      std::cout << " [" << iter->fir          r->second << "], ";
    }
    std::cout << std::endl;
  }

  // std::cout << "now call                              << std::endl;
  computeData(2*i, data[i]);

  //std::cout << data[j].first << ",              ond << std::endl;

  if (data[i].first < 0)
  {
    std::cout << "FATAL ERROR! This should not happen. In line 49 of file!";
    MPI_Abort(MPI_COMM_WORLD, 0);
  }

  value += i;

  if (debug_output_1)
  {
    if (i % 1000 == 0)
    {
      int rankNo = 0;
      MPI_Comm_rank(MPI_COMM_WORLD, &rankNo);
      std::cout << "i: " << i << ", current value: " << value << " (on rank " << rankNo << ")" << std::endl;
    }
  }
}
```

Verbosity switches
getting rank
Why would you need a logging library?
Manually traversing data structure
Logging errors and abort
Output commented out
Output only in some iterations

University of Stuttgart
Cluster of Excellence in Simulation Technology

Thank you!

Benjamin Maier

e-mail Benjamin.maier@ipvs.uni-stuttgart.de
phone +49 (0) 711 685-88247

University of Stuttgart
Institute for Parallel and Distributed Systems
Universitätsstraße 39
70569 Stuttgart