Yet Another Fast Log

yafl is yet another complete technique to fastly: write, run and read log messages

Background

Applications either do their job or log messages about the job, but not both
Natural logging mechanism
Easy to log a message
Minimum constraints on the message itself
Good logging mechanism
Logs Everything
Fast
Usable
Works for the developer and not vise versa

Naturally Written Log Messages @ ${SOURCE}/main.cpp

```
LOG_DEMO_INFO( "messages per second: ", messegesPerSecond, "; calculation time (us):", fastCount, "; total messages:", s1.getCallCount() );
LOG_DEMO_DEBUG( "this text is mapped to unsigned long" );
LOG_DEMO_WARNING( "variables must be converted to unsigned long too:", v1, v2, v3 );
```

Applicability

Embedded
Critical mission areas
No strings as part of the logic (see limitations)
Where logs are part of the defined output of the program

Preprocess

Preprocess

Generates "binary" source files
Generates map file
Adds “metadata”
Keeps the generated code in the correct line (!)

Generated Source Code @ ${BINARY}/main.cpp

```
yafl::log::g_logQ.push(yafl::log::Log(0xe642399bcaccbb09,__LINE__,0x00000000ababab03,pthread_self(),0xbeef000000000000,0xbeef000000000001,messegesPerSecond,0xbeef000000000002,fastCount,0xbeef000000000003,s1.getCallCount() ,0xcdcdcdcdcdcdcdcd));
yafl::log::g_logQ.push(yafl::log::Log(0xe642399bcaccbb09,__LINE__,0x00000000ababab00,pthread_self(),0xbeef000000000004,0xbeef000000000005,0xcdcdcdcdcdcdcdcd));
yafl::log::g_logQ.push(yafl::log::Log(0xe642399bcaccbb09,__LINE__,0x00000000ababab02,pthread_self(),0xbeef000000000006,0xbeef000000000007,v1,v2,v3 ,0xcdcdcdcdcdcdcdcd))
```

Map File

Contains information about the source file itself
Maps between unsigned long to a human readable text

Map File @ ${BINARY}/main.cpp.map

```
'0xe642399bcaccbb09':["/home/ran/work/yafl/yafl/logger/demo/main.cpp",{
'0xbeef000000000000':"Demo",
'0xbeef000000000001':"messages per second: ",
'0xbeef000000000002':"; calculation time (us):",
'0xbeef000000000003':"; total messages:",
'0xbeef000000000004':"Demo",
'0xbeef000000000005':"this text is mapped to unsigned long" ,
'0xbeef000000000006':"Demo",
'0xbeef000000000007':"variables must be converted to long too:",}],
```

Generate

Compile

Executable

Runtime

Runtime

Messages are copied
A dedicated thread buffers the messages
Messages are wittren to a file in batches
Result: extremely low footprint on the working thread

Compile

Errors points to the generated code and not to the original code
Errors in the log message itself are strange (but easy to get used to)
See Improvments

Benchmark

Runtime -
Results are mostly affected by I/O operations
4 CPUs
Intel(R) Core(TM) i5-7600K CPU @ 3.80GHz
Ubuntu x86_64
8Gb Ram
Without I/O: ~5.5 messages per second
With I/O: ~2.6 messages per second
There are tons of issues to improve or check - none of them was done
thread-queue (locking, smarter multithread support, internal queue type)
message construction (easily can be constexpr)
time indication for each message (can avoid system calls (?))
smarter I/O
more? (yes!)

Compile time -
Not tested, surly can be improved

Binary Log File (od -v -A n -t x8 -w200 fast.log)

```
0000016c2dfaeac5 4ea57b2b5505151a 000000000000002e 00000000ababab01 00007f0d67874740
beef000000000003 beef000000000004 0000000000000001 beef000000000005 cdcdcdcdcdcdcdcd  <snip>
```

Cmake

Cmake has a crucial role in binding it all together.
It keeps the usual process untouched:
code -> compile -> run -> analyze
each file is a custom project
for each file a '<file>.gen.sh' is created to generate the code and map dependencies are enforced
See https://github.com/regevran/yafl.git for detals.

Read

Improvements

Faster Runtime
Faster Pre Compile
Faster Post Processing
yafl-grep
yafl-tail
yafl-etc
IDE integration (compiled code is not the written code)
Enable only in partial parts of the code (mix mode with other logs)

Reader

Currently a python script that just translate
Can be a compiled code.
May have few readers types: see improvements

Reader

Limitations

Dynamic strings cannot be logged (see applicability)
All types must be converted to long
No “recursive” logs:

```
cout << a;
ostream& operator << (ostream& o, const A& a) { o << a.b;}
ostream& operator << (ostream& o, const B& b) { o << b.c;}
etc.
```

Output

Text Log File

```
26/07/2019 14:11:55.566|/home/ran/work/yafl/yafl/logger/demo/main.cppl38|INFO|0x00007f0d67874740|Demo|messages per second:  200000; calculation time (us): 5; total messages: 1############
26/07/2019 14:11:55.566|/home/ran/work/yafl/yafl/logger/demo/main.cppl39|DEBUG|0x00007f0d67874740|Demo|this text is mapped to unsigned long ##################
26/07/2019 14:11:55.566|/home/ran/work/yafl/yafl/logger/demo/main.cppl40|WARNING|0x00007f0d67874740|Demo|variables must be converted to long too: 100200300###############
```

Known Issues (bugs)

Commas cannot be added to string messages (need to write better m4 code)

Contact

Ran Regev
regev.ran@gmail.com
@ran_regev
linkedin.com/in/ran-regev-8b57386
+972-50-3891316
https://github.com/regevran/yafl.git