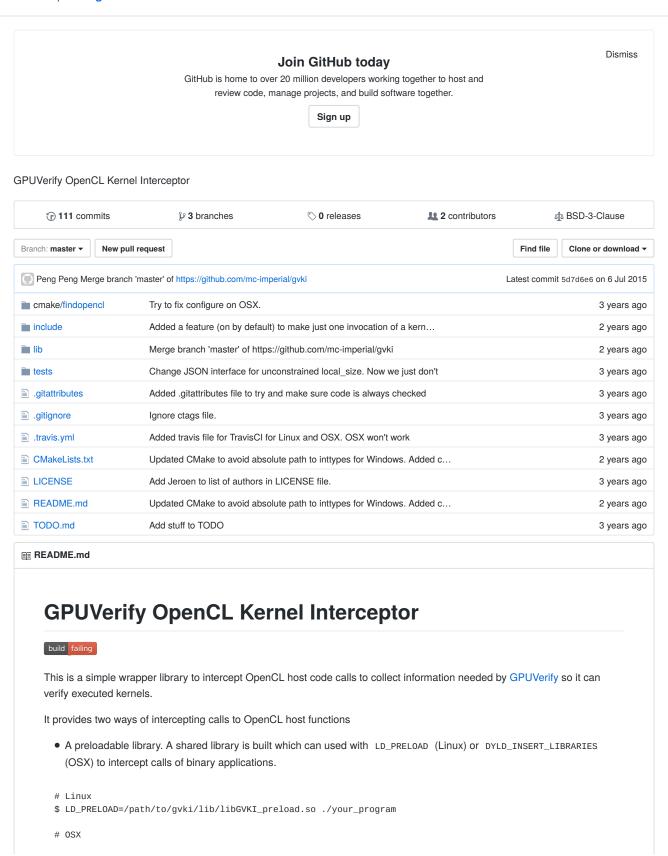
mc-imperial / gvki



1 of 3 9/20/17, 3:47 PM

```
$ DYLD_FORCE_FLAT_NAMESPACE=1 DYLD_INSERT_LIBRARIES=/path/to/gvki/lib/libGVKI_preload.dylib
./your_program
```

"Macro library". For systems that do not support pre loadable libraries we also provide a header file that can be included
in your application to rewrite all relevant calls to OpenCL host functions to calls into our interceptor library
(lib/libGVKI_macro.a) which you then must link with afterwards.

Add #include "gvki_macro_header.h" into your source files just after your include of the OpenCL header e.g.

```
#include <CL/OpenCL.h>
#include "gvki/gvki_macro_header.h"
int main(int argc, char** argv)
{
...
}
```

Then compile your application linking the interceptor library

```
$ gcc -I/path/to/gvki_src/include/ your_application.c lib/libGVKI_macro.a -o your_application
```

Building

Requirements

- CMake >= 2.8.7
- OpenCL header files
- OpenCL library (needed for testing only)
- Python >= 2.7 (needed for testing only)

Linux/OSX

```
$ mkdir gvki
$ git clone git://github.com/mc-imperial/gvki.git src
$ mkdir build
$ cd build
$ cmake -DENABLE_TESTING:BOOL=ON ../src/
$ make
```

Note if you don't have a working OpenCL implementation on your system set ${\tt ENABLE_TESTING}$ to ${\tt OFF}$.

Windows

- 1. Download msinttypes and put it somewhere on your machine: https://code.google.com/p/msinttypes/
- 2. Clone the gvki repository
- 3. Now run cmake-gui and set the source directory to the git repository and the build directory to anywhere you want (preferrably not the git repository)
- 4. Set MSINTTYPES_DIR to be the directory of your msinttypes download that contains inttypes.h.
- 5. Click the Configure button and select the generator you want to use (e.g. Visual Studio 12 2013)
- 6. CMake will try to configure. It is likely that CMake will not be able to find the OpenCL header files and libraries. If it does not you should manually set <code>OPENCL_INCLUDE_DIRS</code> (required) and <code>OPENCL_LIBRARIES</code> (only needed if you want to do testing) in the <code>cmake-gui</code> interface. Once you've done that press the configure button again until it succeeds.

2 of 3 9/20/17, 3:47 PM

- 7. Press the Generate button.
- 8. You can now build the project using the system relevant to the generator you chose. If you chose Visual Studio there will be a .sln file in the build directory you can open.

Testing

Run the test target (ENABLE_TESTING must be set to true when configuring with cmake)

\$ make check

Output produced

When intercepting a <code>gvki-<N></code> directory is created where <code><N></code> is the next available integer. The location of this directories can be controlled using <code>gvki_Root</code>. If <code>gvki_No_Num_DIRS</code> si specified then numbered directories are not created and instead everything is logged into <code>gvki_Root</code> which must not already exist.

The directory contains the following

- log.json file should which contains information about logged executions.
- <entry_point>.<M>.cl files which are the logged OpenCL kernels where <entry_point> is the name of kernel and
 <m> is the next available integer.

An example invocation of GPUVerify on the logged kernels is

\$ gpuverify --json gvki-0/log.json

Special environment variables

Setting various environment variables changes its behaviour

- GVKI_DEBUG Setting this causes debug information to be sent to stderr during interception.
- GVKI_ROOT is the directory that gvki-* directories are created in. If not set the current working directory is used.
- GVKI_LOG_FILE Setting this to a valid file path will cause logging messages to be written to a file in addition to the normal stderr output.
- GVKI_NO_NUM_DIRS Setting this causes GVKI_ROOT to be used as the directory for logging files instead of using gvki-*.

3 of 3 9/20/17, 3:47 PM