Logging in C++ with google logging library

Info: See http://google-glog.googlecode.com
Author: Chad Skeeters
Date: 13-Feb-2013
Description: This document outlines the use of google logging library

ABSTRACT

This document describes the facilities in google logging library that could be used for a c++ project.

Using a logging library can assist developers with debugging code and help maintenance personnel with troubleshooting systemic problems. Often c++ developers use `printf` or `cout` statements to check for proper code flow and to debug issues, but then they have to be removed later. Many logging facilities support *logging levels* that enable logging at a specified level or above without re-compilation.

Source Code Example

```
#include <glog/logging.h>

int main(int argc, char *argv[])
{
  //FLAGS_stderrthreshold=google::INFO;
  //FLAGS_logtostderr = 1; // disabled logging to a file.
  //FLAGS_minloglevel = google::WARNING; // the default is INFO
  //FLAGS_v = 2;
  google::InitGoogleLogging(argv[0]);

  LOG(INFO) << "Something is happening.";
  LOG(WARNING) << "Warning something might not be setup right.";
  LOG(ERROR) << "Error has occurred, but trying to continue";
  LOG(FATAL) << "Program is dieing because...";

  VLOG(1) << "Something is happening";
  VLOG(2) << "Something detailed is happening";
  VLOG(3) << "Something very detailed is happening";
}
```

> Note
>
> no `std::endl` required.

Logging Types

Logging is divided into two types, each with levels. The logs with INFO, WARNING, ERROR, and FATAL should be useful for administrators and users.

Verbose Logging should be made useful developers and maintainers tracking the programs flow through algorithms.

Output

If you call `google::InitGoogleLogging(argv[0])`, the logs go to stderr and the log file in `/tmp/`. Otherwise, logging will only go to stderr.

The name of the program (passed in `argv[0]`) is used in the file name of the log files that are created in `/tmp/`. Unless otherwise specified, glog writes to the filename:

```
/tmp/<program name>.<hostname>.<user name>.log.<severity level>.<date>.<time>.<pid>
(e.g., "/tmp/hello_world.server.chad.log.INFO.20080709-222411.10474")
```

In addition to these log files, symbolic links are created to make it easier to see the last longs. It would be named `/tmp/hello_world.INFO` for the example above.

Configuring

What gets logged can be configured by:

- Flags set in the source code (uncomment lines in the example above)
- Command line arguments provided to the application that set the flags in the example above. (See google flags)
- Environment variables set before the program starts

Most flags documented as a command line parameter can be used as a `FLAGS_` program variable or as a `GLOG_` environment variable.

> **Note**
>
> The command line arguments don't work out of the box without google flags, or some other command line argument processor.

Standard Output

By default, `ERROR`, and `FATAL`, are print to stderr in addition to being output to the logfile. To change this default level, the `FLAGS_stderrthreshold` variable can be set to 0 for `INFO`, 1 for `WARNING`, 2 for `ERROR`, and 3 for `FATAL`. The `GLOG_stderrthreshold` environment variable can be used the same way as shown in the example below.

```
GLOG_stderrthreshold=0 ./hello_world
```

Log Levels

An administrator may decide that they don't want to clutter the files with INFO level logs. To change the logging leve, run the program with the environment variable `GLOG_minloglevel` set to 0 for `INFO`, 1 for `WARNING`, 2 for `ERROR`, and 3 for `FATAL`. Since it's a minimum, setting it to 1 for `WARNING` would include `ERROR` and `FATAL`, but exclude `INFO`.

```
GLOG_minloglevel=1 ./hello_world
```

If the FLAGS_minloglevel flag is uncommented, it will set a default that can be overridden by a argument or environment variable.

Verbose Log Levels

Verbose logging can be enabled globally via the `FLAGS_v` flag or the `GLOG_v` environment variable. It can also be enabled on a per-module basis using the `GLOG_vmodule` environment variable.

```
GLOG_vmodule=processor=2 GLOG_v=1 ./hello_world
```

This will enable any `VLOG(2)` statements in `processor.h` or `processor.cpp`.

Installation

The library and development header can be downloaded on google code.

Read the INSTALL file for details, but this should work

```
sudo make install
```

> **Note**
>
> There are also RPM files on google code if you look for outdated files.

Compiling your programs

When you link your program you must pass the `-lglog` flag. You also use this:

```
pkg-config --libs libglog
```