Glogging

11/05/2018

LOG(INFO) << "A" << "B" ...

GLOG_INFO=0
GLOG_WARNING=1
GLOG_ERROR=2
GLOG_FATAL=3

LOG-Message (__FILE__, __LINE__, GLOG_INFO).stream()

stream() → std::ostream& stream() { data_->stream_; }

LogMessage ~~* LogStreamBuf streambuf_~~

→ LogMessageData* data_;

std::ostream std::streambuf

LogStream
data_ → stream_ :

LogStream ⇔ LogStreamBuf streambuf_;

rdbuf(&)

LogMessageData → LogStream

char* buffer

char message_text_[LogMessage::kMaxLogMessageLen]

LogStream → std::stream

LogStream(char* buf, int len, int ctr) {
  : std::ostream(NULL),
  streambuf_(buf, len), ctr_(ctr),
  self_(this) {
  rdbuf(&streambuf_);
}

std::streambuf

LogStreamBuf

LogStreamBuf(char* buf, int len)
setp(buf, buf+len-2);
// account for '\n' '\0'
}

LOG(INFO) << "a" << "b";

↓

Create a LogMessage and then return its [stream()], which is a std::ostream, so that we can use its "<<" operator.

When LogMessage destroyed, it will call LogMessage::Flush, which will flush all data to stderr/file.

```
{
  MutexLock l(&log_mutex);
  [(this->*(data_->send_method_))();]
  ++num_messages_[static_cast<int>(data_->severity_)];
}
LogDestination::WaitForSinks(data_);
```

void SendToLog() ⟶ LogDestination::LogToAllLogFiles
                   LogDestination::MaybeLogToStderr
void SendToSyslogAndLog().

```
class GoogleInitializer {
public:
  typedef void (*void_function)(void);
  GoogleInitializer(const char*_, void_function f) {
    f();
  }
};

#define REGISTER_MODULE_INITIALIZER(name, body) \
namespace { \
  static void google_init_module_##name() { body; } \
  GoogleInitializer google_initializer_module_##name(#name, \
      google_init_module_##name); \
}
```

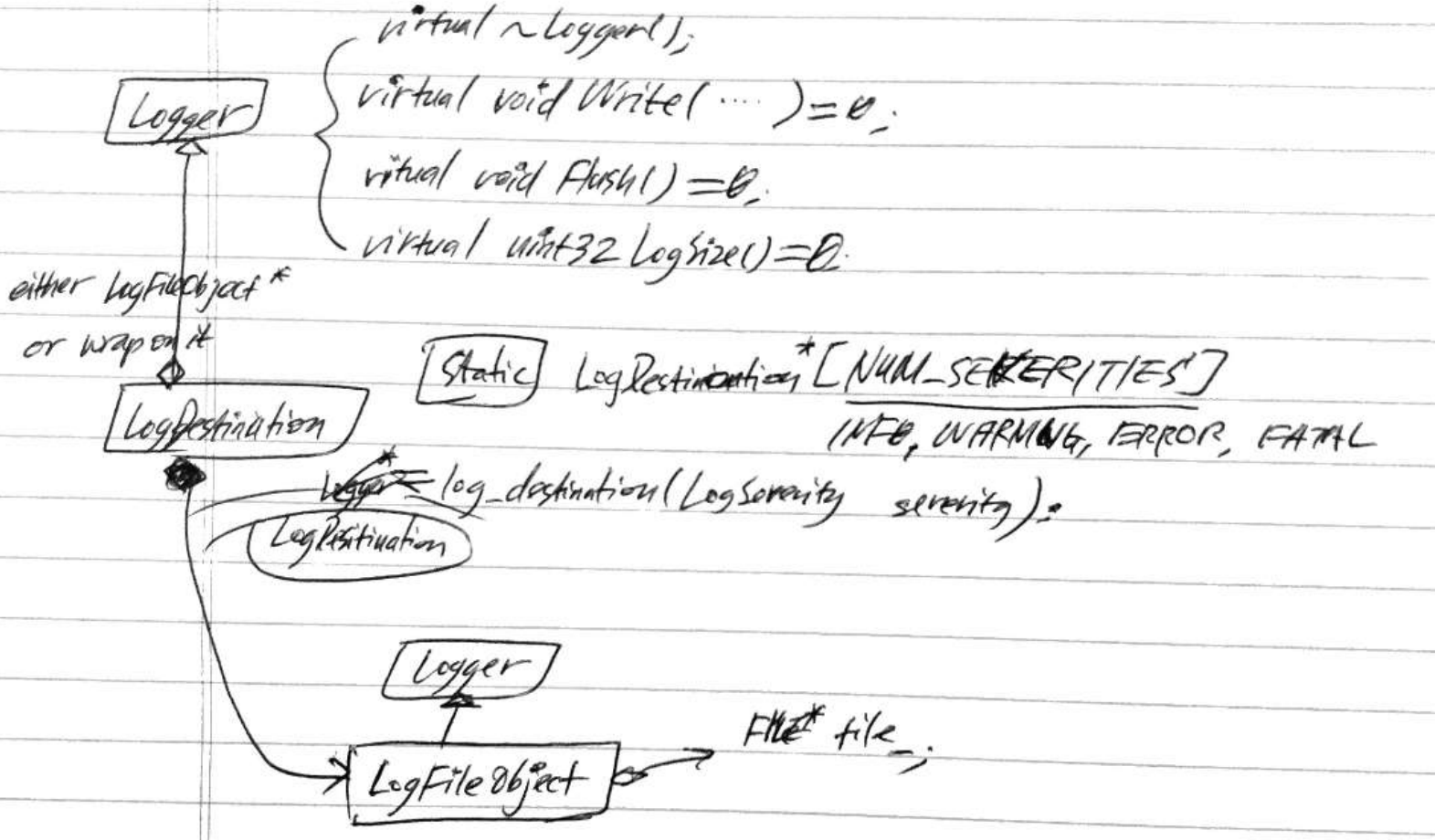
构造函数赋值

```
#define DECLARE_VARIABLE(type, shorttype, name, tn) \
  namespace fL##shorttype { \
    extern GOOGLE_GLOG_DLL_DECL type FLAGS_##name; \
  } \
  using fL##shorttype::FLAGS_##name

#define DEFINE_VARIABLE(type, shorttype, name, value, meaning, tn) \
  namespace fL##shorttype { \
    GOOGLE_GLOG_DLL_DECL type FLAGS_##name(value); \
    char FLAGS_no##name; \
  } \
  using fL##shorttype::FLAGS_##name
```

← 从namespace中导出

```
#define DECLARE_bool(name) \
  DECLARE_VARIABLE(bool, B, name, bool)
#define DEFINE_bool(name, value, meaning) \
  DEFINE_VARIABLE(bool, B, name, value, meaning, bool)
```




```
LogDestination(LogSeverity severity, const char* base_filename)
: fileobject_(severity, base_filename), logger_(&fileobject_)
```