logsys v2

Steven Dake
October 2008

Major Goals

- Flight recorder of logs and events for analyzing field failures (segfaults)
- High performance non-display tracing system
- High performance display logging system
- Maintain most features of logsys v1
- Sanitize API where possible
- Simplify/remove locking implementation

Flight Recorder

- Data block stored at segmentation fault or assertion which maintains chunked circular array of events leading up to failure
  - Use case: service engine fails, but the backtrace doesn't provide enough information to analyze what went wrong 20 events ago...
- Tracing events (high volume) are separate constructs from logging messages (low volume)

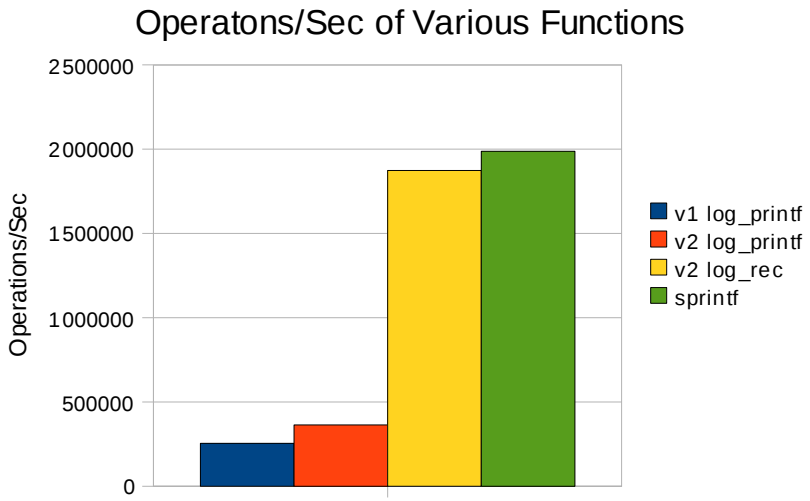
Flight Recorder – Information Recorded

- Subsystem name
- Record Identifier
- C Filename
- C Function name
- File line number
- Up to 64 variable length arguments

Flight Recorder – Special Cases

- Logging messages recorded to flight recorder as special record identifier with log message as argument
- Entering and leaving of functions recorded to flight recorder as special record identifier
- Future possibility for tracing levels to be specified in the record identifier

Logsys v2 Performance



sprintf prints a log message to a buffer with the argument “recordA”.
log_rec records an event with the argument “recordA” through the flight recorder.
log_printf both v1 and v2 does not display the log output, but executes all other code paths.

Tested on 2ghz Thinkpad T60.

Logsys v2 Usage API

- log_rec (REC_IDENT, argumentN ptr, argumentN length, (any other arguments), LOG_REC_END);
- log_printf (level, printf_format_string, arguments);
- ENTER();
- LEAVE();

log_printf Formatting

- A formatting string can be specified during initialization to control display of logging information
  - %s subsystem, %n function name, %f filename, %l fileline, %p priority, %t timestamp, %b buffer
  - A number between % and the control character specifies the width of the field
- Example formatting string “[%6s] %b”
  - [MAIN] Corosync Executive Ready.

API Changes

- LOGSYS_DECLARE_SYSTEM, logsys_init take two additional arguments:
  - format – The formatting string used for the logging output
  - flt_size – Size of the flight recorder buffer in 32 bit words
- The following log modes are removed:
  - LOG_MODE_DISPLAY_PRIORITY, LOG_MODE_DISPLAY_TIMESTAMP, LOG_MODE_DISPLAY_FILELINE, LOG_MODE_BUFFER_ BEFORE_CONFIG, LOG_MODE_FLUSH_AFTER_CONFIG, LOG_MODE_SHORT_FILELINE
- The following log modes are added
  - LOG_MODE_FORK – The application will later call logsys_fork_completed()
  - LOG_MODE_THREADED – Operate in a threaded non-blocking mode
- New API logsys_fork_completed() to indicate any runtime configuration is completed and any fork to detach the tty has occurred and the worker thread can be initialized.

Conclusion

- Trace events and log events have separate meanings. They are separated in the API.
- Use log_rec(), ENTER(), LEAVE() for super high performance failure analysis tracing while leaving it in the source tree at all times for developers to use.
- Use log_printf() for informational messages that the user can understand.
- The logsys v2 code is superior in terms of performance and functionality.
- If you find log_rec too hard to use, full v1 semantics of log_printf are supported.