ErrorLogger

James T. Linnemann
Michigan State University
May 27, 1998

Pre-History

- Tim Cox implemented Run I functionality in C++
  - needs of Run II L2, L3 folded in
  - only dynamic allocation was string handling
    - ②could easily make static, compatible with string stream
- not entered in CVS, testing not final
  - complaints about insufficiently “objectivite” thinking
  - Tim left for another job

Recent History

- JTL discussed with Wyatt, Amber
  - authorized to negotiate with Zoom for D0
    - ②Mark Fischler, Walter Brown (CD)
    - ②Liz Sexton-Kennedy (CDF)
  - Run I functionality judged sufficient
  - NOTE: This was static

L2 Requirements (L3 less restrictive)

- static, or rather dynamic only during initialization
- static strings (C 0-terminated strings)
- layering:
  - logging via buffer
  - relay of messages to online “alarm” monitor
- C++ probably now acceptable
- I/O only via defined error message devices
- limited external libraries (e.g. beware STL)
  - system utilities: no operating system!

User Interface

```
log << ELerror << "Too much energy" << "E = " << total Energy << endl;
```

- stream syntax (rather than function call)
- log is logname
- ELerror is severity
- 1st 20 characters of 1st string is ID
- endl is separator of error messages
  if forgoten, next severity separates
- subroutine acquired by extra call, or by partial traceback

```
%ERLOG-w  Too much energy
        E = 834.750032
        processname CTCDRVmodule CTCTRKsubr
        10-Jul-1999 14:49:03 run=234 event=543
```

Severity Levels

Severity	Symbol	Name	Intent
ELzeroSeverity	--	--	use only in setup
ELincidental	..	..	flash on a screen
ELsuccess	-!	SUCCESS	report a milestone
ELinfo	-i	INFO	information
ELwarning	-w	WARNING	warning
ELwarning2	-W	WARNING!	more serious warning
ELerror	-e	ERROR	error detected
ELerror2	-E	ERROR!	more serious error
ELnextEvent	-n	NEXT	advise to skip event
ELsevere	-s	SEVERE	future results suspect
ELsevere2	-S	SEVERE!	more severe
ELabort	-a	ABORT!	suggest aborting
ELfatal	-F	FATAL!	strongly suggest aborting!
ELhighestSeverity	!!	!!	Use only in setup

Framework interface: constructor

```
ErrorLogger (   void (*context) (string & run, string & event),
                ostream & logStream,
                ostream & screenStream = 0 );
```

- defines “context” function to return event ID
- defines log and screen devices
  to real streams, buffers, devices, or null

Threshold:
Min Severity for Function w/ Default

log.setThresholdLog (ELerror) ELwarning

log.setThresholdScreen (ELinfo) ELincidental

log.setThresholdSave (ELwarning2) ELwarning

log.setThresholdStats (ELerror2) Elerror

log.setTraceThresholdLog(ELerror2) Elerror

log.setTraceThresholdScreen(Elerror) Elerror

Limit:
Max count for message

log.setLimit(“*”,5,3);

5 max to Log device, 3 max to screen

log.setLimit(“this error id”,10,20)

Extra Information

Frame:

ErrorLogger::setProcess(“RECO Linux 23”);

ErrorLogger::setModule(“Package Name”);

User:

ErrorLogger::setSubroutine(“My_Subroutine”);

Short Term Memory

- ErrorHighest = Log.checkSeverity();
- log.severityCount(ELerror);
- log.severityCount(Elsevere,ELfatal);

- Log.flushSaved(buffer);

  buffer is pointer to output area

  can use to attach messages to an event

ErrorLogger Summary

- log.summary();

  to log

- log.summary(s);

  to stream s

  count by ID, module, severity

  three examples of each with event ID

- log.grandSummary(s);

  adds aggregate counts

Run Boundaries

- Log.clearSummary();
- log.wipeSummary();

  including aggregate counts

Differences from ERRMSG

- C strings, C++ syntax
- Stream syntax (nice for formatting)
  - beware: string stream hard to re-initialize
    - ②better to just let go out of scope
- No explicit ID; subroutine optional
- more severity levels
- Longer message (more info, e.g. date/time)
- bigger summary, including examples
  - classified by Module, severity, not just ID
- few GET routines

Outstanding Questions

- Implementation issues for online (libraries, dynamic memory, I/O, STL…)
- case insensitivity for ID's?
- message verbosity
- summary verbosity?
- multiline messages and boundaries in buffers
- behaviour when initial table allocation hit
- interaction of * and specific message limits
- length on log.flushSaved?
- Log.wipeAllButLimits()?