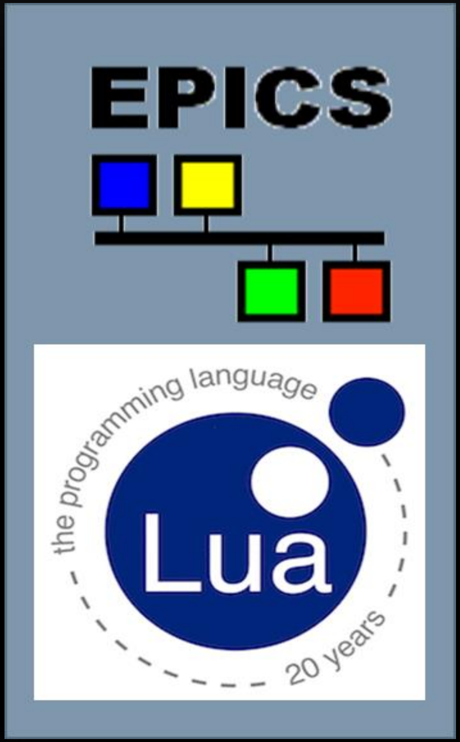
THE EPICS LUA SCRIPT RECORD

Jeff Hill

LANSCE

THE EPICS LUA SCRIPT RECORD – OUTLINE

- Lua, a Brief Introduction (review)
- EPICS Integration of Lua milestones
- Lua Record Goals
- Lua Record Design
- Lua Record Robustness Features
- Data Access Interface to 3rd Party Data
- DBF_VARIENT database type
- Lua Record Pitfalls
- Conclusions



THE EPICS LUA SCRIPT RECORD – LUA A BRIEF INTRODUCTION (REVIEW)

- Lua *embeddable* language was created in 1993
  - By members of the Computer Graphics Technology Group (Tecgraf) at the Pontifical Catholic University of Rio de Janeiro, in Brazil.
- "Lua" (pronounced **LOO-ah**) means "Moon" in Portuguese
- Interpreted, compiled at load-time to byte-code
- A mixture of C-like and Pascal-like syntax
- Dynamic typed, automated conversion between string and numberic types
- Efficient virtual machine execution, small footprint, incremental garbage collection, easily interfaced with C code
- Liberal MIT license
- Some negatives also, see my talk at Michigan EPICS meeting
  - In particular, variables are globally scoped by default

THE EPICS LUA SCRIPT RECORD – EPICS INTEGRATION OF LUA MILESTONES

- Lua 5.2.3, the current release, embedded inside of EPICS base
  - Built by the EPICS build system
  - This is the current released version of Lua
    - It has the upgraded support for integer primitive types

THE EPICS LUA SCRIPT RECORD
– EPICS INTEGRATION OF LUA MILESTONES

- Lua based subscription filtering in the CA server
  - Event queue is order correct
  - Based on C++ 11 shared pointer
    - Subset of boost included in EPICS base supporting prior compilers

THE EPICS LUA SCRIPT RECORD – EPICS INTEGRATION OF LUA MILESTONES

- Lua based subscription filtering in the CA server
  - Snap-in interface for LANSCE timed-and-flavored subscription filters
  - Filters specified as channel name postfix
    - Invoking Lua methods supplied when the IOC boots
  - Each client attaching to the server
    - Instantiates an independent Lua context

THE EPICS LUA SCRIPT RECORD – EPICS INTEGRATION OF LUA MILESTONES

- Alternative EPICS SHELL
    - In contrast, a fully functionality scripting language
        - Powerful libraries, built-in and community
- An environment well proven for use in
    - Configuration
    - Scripting
    - Rapid-prototyping

THE EPICS LUA SCRIPT RECORD
– EPICS INTEGRATION OF LUA MILESTONES

- EPICS IOC shell can invoke, and pass arguments to, Lua scripts
- Lua scripts can invoke, and pass arguments to
  - Any of the commands registered into EPICS IOC shell
  - We can, for example, instantiate records within a Lua for loop

THE EPICS LUA SCRIPT RECORD – LUA RECORD GOALS

- Currently we have two computational record-level building block components
  - EPICS calc record
    - Excellent rapid prototyping, but limited functionality
  - EPICS subroutine record
    - Excellent efficiency, but possibly less popular for rapid prototyping
- A new Lua based record might provide
  - Comprehensive functionality set
  - A reasonable compromise runtime execution efficiency
  - The rapid prototyping we depend on with the calc record
    - Runtime changes via CA puts to lua record fields
- And, we hope that the heavy lifting might come for free with Lua

THE EPICS LUA SCRIPT RECORD – LUA RECORD DESIGN

- Independent Lua context for each Lua record – this *is* somewhat expensive but …
  - They are not making small memory chips any-longer
  - Sometimes its best not to share …
    - Application specific Lua heap usage has a global impact on performance
    - Global variables sharing between Lua records
      - Perhaps its just smart to avoid software dark alleys
        - We don't like it when a new Lua record breaks another record that was installed 10 years ago
  - Single threaded access to the Lua state
    - No MUTEX locking wrapping of Lua C library calls
      - Less runtime overhead

THE EPICS LUA SCRIPT RECORD – LUA RECORD DESIGN

- Independent Lua context for each Lua record
    - Nevertheless, we will need to share some common infrastructure
        - Lua tables, function, libraries, class libraries
    - A site or application specific assortment of startup scripts is needed
        - To initialize each record's private Lua context

THE EPICS LUA SCRIPT RECORD – LUA RECORD DESIGN

- The *file name* of a configuration script is specified by the LUAS field
- This startup scripts initialize the Lua context instantiating supporting infrastructure
  - Instantiating any Lua functions and libraries needed
  - Instantiating any Lua data, tables, objects needed

THE EPICS LUA SCRIPT RECORD – LUA RECORD DESIGN

- The LUAS field specified configuration script runs when
    - The record is initialized
    - Also whenever a CA client modifies the LUAS field
        - The Lua context is destroyed
        - A new Lua context is created
        - The LUAS field specified configuration script is run against the new Lua context
        - The PACT field is restored to FALSE
            - More on this later

THE EPICS LUA SCRIPT RECORD – LUA RECORD DESIGN

- The LUAE field specifies the Lua equivalent of the CALC expression
- This expression is executed
  - When the record is processed
  - Its result is placed in the record’s VAL field

THE EPICS LUA SCRIPT RECORD – LUA RECORD DESIGN

- The LUAE field expressions are wrapped with a Lua function
  - So they can access the lua record’s input fields, passed as input arguments
  - "function (a, b, c, d, e, f, g, h, i, j, k, l) return %s; end“
  - The expression in the LUAE field is substituted for %s in the quoted string above
  - The a, b, c … l are the values of the record’s similarly named input link fields
    - Input fields are
      - Read each time the record is processed
      - Pumped onto the Lua stack
      - Become input arguments for the LUAE field’s Lua expression

THE EPICS LUA SCRIPT RECORD – LUA RECORD DESIGN

- A new Lua expression is compiled by Lua when
  - The record initializes
  - Also whenever clients modify the LUAE field
  - New Lua code causes PACT field restoration
    - Set to FALSE

THE EPICS LUA SCRIPT RECORD – ROBUSTNESS FEATURES

- Lua protected call library function is used
  - To invoke the LUAE and the LUAS specified Lua code
  - Therefore, Lua exceptions are caught before returning Lua code into C code
- This implies that if a user Lua code throws an uncaught exception
  - Then, debug trace back messages are printed on the command line
- Processing of the Lua record is disabled
  - PACT field is left in true state
    - Effectively disabling the record
  - The record is also placed in invalid alarm state
  - Therefore, the stack-trace message is printed only once
    - CPU is not consumed repetitively running an exception handler

THE EPICS LUA SCRIPT RECORD – DATA ACCESS INTERFACE TO 3RD PARTY DATA

- Data Access
  - A Data Type extension mechanism
    - For indexing and traversing 3rd party hierarchical data
  - C++ pure virtual base class, and associated support library
  - It can be used to interrogate data coming from almost any source
  - Comparable to device support, record support, asyn, streams …
    - With device support *system programmers* interface 3rd party devices
    - With Data Access *system programmers* interface 3rd party data sources
  - Application developers use newly interfaced data types
    - They are not required to know about low level Data Access interfaces

THE EPICS LUA SCRIPT RECORD – DBF_VARIENT DATABASE TYPE

- DBF_VARIENT type contains three C++ 11 shared_ptr objects
  - Pointer to a Data Access Index interface
  - Pointer to a Data Access Mutator interface
  - Pointer to lifetime management interface
- The DBF_VARIENT type is an extension mechanism for 3rd party data
  - Is uses as the value field of advanced record types
- Lua records are also interfaced to the Data Access Index interface in the DBF_VARIANT
  - Using the Lua table index extension mechanism
  - Lua can index any of the properties in hierarchical 3rd party data, for example
    - "a.processVariable.alarm.condition.status"

THE EPICS LUA SCRIPT RECORD – LUA RECORD PITFALLS

- The Lua garbage collector runs incrementally however …
    - Your record will run much more efficiently
        - If you don’t allocate, and subsequently free, Lua heap resources
            - Each time the record is processed
        - Use the Lua stack instead to allocate dynamic memory during record processing
- Lua variables are globally scoped by default

THE EPICS LUA SCRIPT RECORD – CONCLUSION

- Lua *embeddable* scripting language capabilities have been integrated into EPICS
  - CA server event queue filtering
  - Lua based IOC shell
  - Lua record
    - An upgrade for the CALC record
      - With a comprehensive feature set provided by Lua!