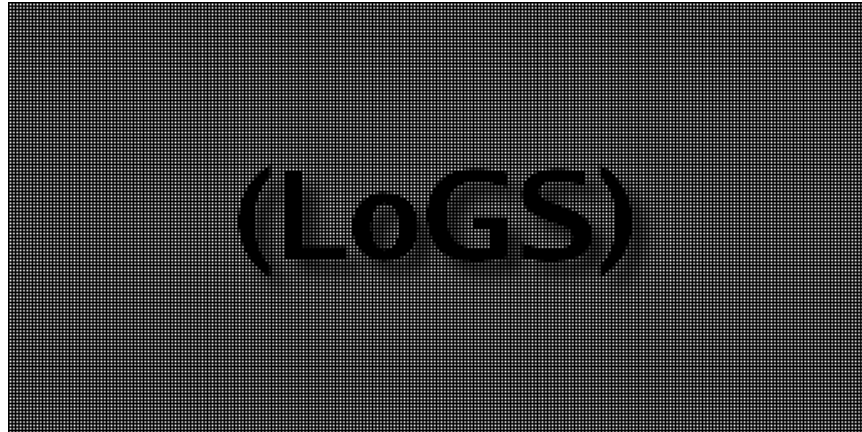


LoGS – yet another log analysis tool?



James E. Prewett OpenPGP key: pub 1024D/31816D93 2003-01-06
Fingerprint=F618 949F 5FB8 918E 8378 8C1F BFFC DDC6 3181 6D93



Motivation for LoGS

- Like Logsurfer (<http://www.cert.dfn.de/eng/logsurf/>), but
 - easy to write rules that create rules that create rules ...
 - a more efficient ruleset design
 - a powerful programming language
 - fewer external programs will be needed
 - small (core) codebase (0.0.1 is about 700 lines of code!)
 - easy to extend



Inspiration for LoGS

- Risto got a lot of things right with Simple Event Correlator (<http://www.estpak.ee/risto/sec/>)
- The Supermon monitoring package. I like their reasoning for using s-expressions to represent data!
- My co-author, Dave, can be a real jerk. (But that's why we love him :)
- Manny Rettinger taught me to listen to my machines.



Components

- Messages – single chunks of data
- Rules — match messages & trigger actions
- Contexts — store multiple messages or just relevant data
- Rulesets —
 - Organize Rules
 - May be used to improve efficiency.



Components continued

- Actions —
 - do things when triggered by:
 - * A rule matching a message
 - * A context exceeding one of its limits
 - Actions can:
 - * create or delete rules
 - * create or delete contexts
 - * create or delete entire rulesets
 - * anything you can write a function to do
 - Actions are: Common LISP functions



LoGS Mainline

```
(defun main ()
  (loop as *message* = (get-logline *messages*)
        while *message*
        do
          (let ((head (head *ruleset*)))
            (when (and *message* head)
              (check-rules *message* *ruleset*))
            (when head
              (check-rule-limits *ruleset*))
            (when (head *contexts*)
              (check-context-limits *contexts*)))))
```



Messages

Message objects consist of:

- a Message.



Rules

Rule objects consist of:

- Name (NEW!)
- Match function
- Delete-rule function
- Timeout
- Actions list
- No-Match function
- No-Delete-rule function
- continue value



Why a match *function*?

- Regular Expressions are not *always* the right choice.
- Certain idioms are simpler to express with code than regexps.
- Bayesian or other techniques may be used?
- It was the only way to shut Dave up! ;)



Simple Rule

```
(make-instance
  'rule
  :match (lambda (message)
            (declare (ignore message))
            t)
  :name 'default-rule
  :actions
  (list
   (lambda (message matches sub-matches)
     (declare (ignore matches sub-matches))
     (print (message message))))))
```



Contexts

Context objects consist of:

- Name
- Timeout
- Max-lines
- Actions list



Using Contexts

```
(make-instance 'context :name 'every-message)
(make-instance
  'rule
  :match (lambda (message)
           t)
  :name 'collect-all-messages
  :actions
  (list
   (lambda (message matches sub-matches)
     (add-to-context 'collect-all-messages
                    message))))
```



Dynamic Rules

Rules in LoGS may be created in response to certain messages.

- rule-before
- rule-after
- rule-head
- rule-tail



Contexts and Dynamic Rules

```
(make-instance
  'rule
  :match
  (lambda (message)
    (cl-ppcre::scan-to-strings
      "xntpd \\[[0-9]\\]: synchronization lost"
      (message message)))
  :name 'find-ntp-problems)
```



Contexts and Dynamic Rules Cont.

```
:actions
(list
  (lambda (message matches sub-matches)
    (make-instance
      'context
      :timeout (+ *now* 3600)
      :name (format ()
                "xntpd~A" (aref sub-matches 0))
      :actions
      (list
        (lambda (context)
          (format t "~A" (aref sub-matches 0)))))))
```



Contexts and Dynamic Rules Cont.

```
(lambda (message matches sub-matches)
  (rule-before
    (make-instance
      'rule
      :match
      (lambda (message)
        (cl-ppcre::scan-to-strings
          (format () "xntpd\\[~A\\]: synchronized to"
                    (aref sub-matches 0))
          (message message)))
      :actions
      (list
        (lambda (message matche sub-matches)
```



```
(delete-context
  (get-context
    (format ()
      "xntpd ~A"
      (aref sub-matches 0)))))))))
```



Ruleset

Ruleset objects consist of:

- Everything in a Rule object
- A doubly-linked-list of rules in the ruleset.



Actions

- Rule/Ruleset actions are Common LISP functions
 - accepts lambda list (message matches sub-matches)
- Context actions are Common LISP functions
 - accepts lambda list (context)



The “Why LISP?” Slide

So, why is LoGS written in LISP?

- I hope that Logsurfer will still be the dominant application written in C. (Logsurfer is back!)
- I wanted a high-level language.
- I wanted to use S-expressions to represent rules; they are free with LISP.
- compilation at runtime (to decent looking assembler with CMUCL!)
 - LISP can be **really** fast



- ... but I don't know LISP that well ...
- CL-PPCRE is often faster than Perl when matching regexps!
- LISP is an Object Oriented language (CLOS) with many niceties not found in other OO languages.
- reader macros (! reader macro in LoGS 0.0.1)
- macros
- interesting code is available
- LISP is **really** cool!

