Streaming regex matching and substitution by the `sregex` library

😊 agentzh@gmail.com 😊

Yichun Zhang (agentzh)
In efficient web servers, request bodies and response bodies are processed in data chunks.
We usually use a *fixed size* buffer even we are processing a much *larger* data stream.
Backtracking regex engines suck.
/abc|ab/ (backtracking)
Thompson's Construction Algorithm comes to rescue!
\[\{abc \mid ab\} \quad (\text{Thompson's Construction})\]
It also supports submatch captures!
\( /a(bc)|a < b)/ \) (Pike's edition)

Diagram:

1. Save \#0
2. \(b\)
3. \(c\)
4. Save \#1
5. \(a\)
6. Save \#2
7. \(a\)
8. Save \#3
DFAs cannot find the beginnings of submatch captures without matching backwards.
\[
\begin{array}{cccc}
\#0 & \#1 & \#2 & \#3 \\
\downarrow & \downarrow & \downarrow & \downarrow \\
/a\, (bc)\, |\, a\, (b)\,/ & (DFA) \\
\end{array}
\]

Step 1: match /a(bc)|a(b)/ to locate \#1 & \#3.

Step 2: match /c(b)a|b(a)/ to locate \#0 & \#2.
I created the sregelex library based on Russ Cox's re1 library.
A non-backtracking regex engine matching on data streams — Read more

bugfix: 8-bit integer overflow was not detected properly in regex not...
sregex is written in pure C.
sregex includes *two* engines: Thompson VM & Pike VM.
\^  \$  \A  \z  \b  \B
.  \c  [\0-9a-z]  [^\0-9a-z]
\d  \D  \s  \S  \h  \H
\v  \V  \w  \W  \cK  \N
ab  a\|b  (a)  (?::a)  a?  a*
a+  a??  a*?  a+?  a{n}  a{n,m}
a{n,}  a{n}?  a{n,m}?
a{n,}?  \t  \n  \r  \f
...
Passing all the related test cases in both the official PCRE 8.32 and Perl 5.16.2 test suites.
#include <sregex/sregex.h>

... 

rc = sre_vm_pike_exec(vm_ctx, pos, len, last_buf, &pending_matched);
The Thompson VM has a simple \textit{Just-in-Time} (JIT) compiler targeting x86_64.
The regex JIT compiler uses DynASM which powers LuaJIT's interpreter.
Still a lot of important *optimizations* to do.
My Nginx C module **ngx_replace_filter** is the *first user* of sregex.
Streaming regular expression replacement in response bodies — Read more

```markdown
replace-filter-nginx-module /
```

bugfix: ignore responses with a non-empty Content-Encoding response h... ...

agentzh authored 3 months ago
location ~ '.cpp$' {
    # proxy_pass ... / fastcgi_pass ...

    # remove all those ugly C/C++ comments:
    replace_filter '/\*\.*?\*/|//\[\^\n\]/*' '' g;
}

# skip C/C++ string literals:
replace_filter "'(?:\\[^n]\|[^'\n])*'" & g;
replace_filter "'(?:\\[^n]\|[^'\n])*'" & g;
replace_filter_max_buffered_size 8k;
Thank you! 😊