"Grep was invented for me. I was making a program to read text aloud through a voice synthesizer. As I invented phonetic rules I would check Webster's dictionary for words on which they might fail. For example, how do you cope with the digraph `ui', which is pronounced many different ways: `fruit', `guile', `guilty', `anguish', `intuit', `beguine'? I would break the dictionary up into pieces that fit in ed's limited buffer and use a global command to select a list. I would whittle this list down by repeated scannings with ed to see how each proposed rule worked."

"The process was tedious, and terribly wasteful, since the dictionary had to be split (one couldn't afford to leave a split copy on line). Then ed copied each part into /tmp, scanned it twice to accomplish the g command, and finally threw it away, which takes time too."

"One afternoon I asked Ken Thompson if he could lift the regular expression recognizer out of the editor and make a one-pass program to do it. He said yes. The next morning I found a note in my mail announcing a program named grep. It worked like a charm. When asked what that funny name meant, Ken said it was obvious. It stood for the editor command that it simulated, g/re/p (global regular expression print)."

From an interview with Doug McIlroy

"Progress on my talking program accelerated dramatically. From that special-purpose beginning, grep soon became a household word. (Something I had to stop myself from writing in the first paragraph above shows how firmly naturalized the idea now is: `I used ed to grep out words from the dictionary.') More than any other single program, grep focused the viewpoint that Kernighan and Plauger christened and formalized in `Software Tools': make programs that do one thing and do it well, with as few preconceptions about input syntax as possible."

Regular expressions

Most of the characters in a pattern "match" themselves; the simplest pattern consists only of these literals

For example, the literal **Elian** matches the following sentences from Jeb Bush's emails (among others):

```
Adriana, we continue to ask for Elian's day in court and that his Miami family meet with him.

I agree with you on how Washington has handled the Elian case.

Monica, I have been outspoken in my opinion of the handling of the Elian matter.

However, I respectfully disagree with the handling of taking Elian out of a home by gunpoint.

In fact, as it relates to Elian, this is the first time that I have agreed with you.

I agree with you about the Elian matter.

The armed extraction of Elian yesterday was inappropriate and unnecessary.
```

Regular expressions

It's useful to see what's going on when a "match" occurs; suppose we're looking for the pattern **Elian**, and the target string is "***I agree with you on how Washington has handled the Elian case.***"

The regular expression engine moves along the target string character-by-character, testing for a match of the first literal in the pattern, the **E** — it fails to match until the 51st position

Once it matches the **E**, it advances to the next literal and sees if the 52nd character in the target string is a match, is an **I** — it will continue in this way until it matches the complete pattern (possibly many times) or runs out of characters in the target string

If the 52nd character hand not been an **I**, the machine would reset back to **E** to start its search again at position 52

The mechanics

At a technical level, you can think of a regular expression as **a (finite) state machine** that changes its state as it processes a string character-by-character according to rules specified by the pattern

Ultimately, the machine determines that the pattern exists within the string it is examining (a match) or it does not; we'll have more to say about this when our patterns get a little more complicated

In this way, you can almost think of a regular expression as a kind of program -- A program that identifies a pattern

Programming Techniques

R. M. McCLURE, Editor

Regular Expression Search Algorithm

KEN THOMPSON
Bell Telephone Laboratories, Inc., Murray Hill, New Jersey

A method for locating specific character strings embedded in character text is described and an implementation of this method in the form of a compiler is discussed. The compiler accepts a regular expression as source language and produces an IBM 7094 program as object language. The object program then accepts the text to be searched as input and produces a signal every time an embedded string in the text matches the given regular expression. Examples, problems, and solutions are also presented.

The Algorithm

Previous search algorithms involve backtracking when a partially successful search path fails. This necessitates a lot of storage and bookkeeping, and executes slowly. In the regular expression recognition technique described in this paper, each character in the text to be searched is examined in sequence against a list of all possible current characters. During this examination a new list of all possible next characters is built. When the end of the current list is reached, the new list becomes the current list, the next character is obtained, and the process continues. In the terms of Brzozowski [1], this algorithm continually takes the left derivative of the given regular expression with respect to the text to be searched. The parallel nature of this algorithm makes it extremely fast.

The Implementation

The specific implementation of this algorithm is a compiler that translates a regular expression into IBM 7094 code. The compiled code, along with certain runtime routines, accepts the text to be searched as input and finds all substrings in the text that match the regular expression. The compiling phase of the implementation does not detract from the overall speed since any search routine must translate the input regular expression into some sort of machine accessible form.

In the compiled code, the lists mentioned in the algorithm are not characters, but transfer instructions into the compiled code. The execution is extremely fast since a transfer to the top of the current list automatically searches for all possible sequel characters in the regular expression.

This compile-search algorithm is incorporated as the context search in a time-sharing text editor. This is by no means the only use of such a search routine. For example, a variant of this algorithm is used as the symbol table search in an assembler.

It is assumed that the reader is familiar with regular expressions [2] and the machine language of the IBM 7094 computer [3].

The Compiler

The compiler consists of three concurrently running stages. The first stage is a syntax sieve that allows only syntactically correct regular expressions to pass. This stage also inserts the operator "." for juxtaposition of regular expressions. The second stage converts the regular expression to reverse Polish form. The third stage is the object code producer. The first two stages are straightforward and are not discussed. The third stage expects a syntactically correct, reverse Polish regular expression.

The regular expression $a(b | c) \cdot d$ will be carried through as an example. This expression is translated into $abc | \cdot d \cdot$ by the first two stages. A functional description of the third stage of the compiler follows:

The heart of the third stage is a pushdown stack. Each entry in the pushdown stack is a pointer to the compiled code of an operand. When a binary operator ("|" or ".") is compiled, the top (most recent) two entries on the stack are combined and a resultant pointer for the operation replaces the two stack entries. The result of the binary operator is then available as an operand in another operation. Similarly, a unary operator ("*") operates on the top entry of the stack and creates an operand to replace that entry. When the entire regular expression is compiled, there is just one entry in the stack, and that is a pointer to the code for the regular expression.

The compiled code invokes one of two functional routines. The first is called NNODE. NNODE matches a single character and will be represented by an oval containing the character that is recognized. The second functional routine is called CNODE. CNODE will split the

Regular expressions

Any character except for `[ \ ^ $ . | ? * + ( ) { }` can be used to specify a **literal**; they match **a single instance of themselves** (the rest serve a role as metacharacters that we'll describe in a second; we'll also describe what to do if your pattern involves one of these)

Again, the metacharacters allow us to specify much more complicated patterns; for example, what if we only want the word "Elian"? or sentences that end in "do." or "do?"

Regular expressions

It's clear that we need a way to express

white space

word boundaries

sets or classes of literals

the beginning and end of a line

alternatives ("war" or "peace")

Metacharacters to the rescue!

Regular expressions

We'll now present some simple metacharacter constructions

^,$,\b to specify positioning

[and] to express character classes (or equivalence classes)

(and), | to define subexpressions and alternatives

*,+,? to indicate multiplicities

Some metacharacters

^ represents the start of a line

^I hope

will match the following lines from Jeb's email in April, 2000

```
I hope you and your family are doing well.

I hope you are having a good weekend.

I hope you understand.

I hope you are having a good day.

I hope she will be given a fair hearing.

I hope you and yours have had a great Easter.

I hope you had a joyous Easter.

I hope you have a joyful and joyous Easter.

I hope we can help.
```

← → C ⟳ 📄 regexper.com/#%5El%20hope ☆ 📷 ≡

REGEXPER

You thought you only had two problems…

▤ Changelog
📄 Documentation
`</>` Source on GitHub

```
^I hope
```

Display

⬇ Download // 🔗 Permalink

●─ Start of line ─ "I hope" ─●

Created by Jeff Avallone // Generated images licensed: (cc) BY

Some metacharacters

^ represents the start of a line

^I will

will match the following lines from Jeb's email in April, 2000

```
I will know by tomorrow.

I will probably sign both if they have the criteria that I have asked for.

I will never give up on my home town.

I will check with Kate Kearney to see what she thinks.

I will tell you that the "wants" always seem to be defined as "needs" and
if someone doesn't say no, there is literally billions of "needs".

I will follow the bill carefully.

I will decide next week.

I will check with the team.

I will review the bill should it make it through the process.

I will leave it up to you all to decide what to do and to respond.

I will carefully review the bill should it make it through the process.

I will share your thoughts with the Judge.

I will give the bill every consideration should it become law.
```

Some metacharacters

$ represents the end of a line

do.$

will match the following lines from Jeb's email in April, 2000. Notice anything odd here?

```
let me know what I can do.

will do.

This is a little late but I will check to see what we can do.

What are the incentives being offered by Orlando?

What is the Senate going to do?

Jerry, what should the legislature do to enhance what existing law doesn't do?
```

Some metacharacters

"." is used to refer to any character and so

  do.$

will match lines that end in "do." or "do?" or even "do9" (should Jeb be typing sloppily one day)

regexper.com/#do.%24

REGEXPER

You thought you only had two problems…

☰ Changelog
📄 Documentation
</> Source on GitHub

do.$

Display   ⬇ Download // 🔗 Permalink

●— "do" —[any character]—[End of line]—●

"Escaping" metacharacters

Putting a backslash \ before one of the special characters [\^$/?*+(){}
lets us include these in a pattern as literals — In technical terms, we
have "escaped" the special meaning of these characters

\$1

will match these lines (now drawn from Jeb Bush's emails from the first
six months of 2000)

```
I have bought $100 worth of scrumptous Girl Scout cookies in the last month!

Next year, we are proposing a one mil reduction and raising the exemption of
for single and joint filers respectively to $100,000 and $200,000.

how could Pru lose $122 million?

My concern is that the bill does not deal with the responsibility part with a
$10,000 insurance requirement.

Dick, we are cutting the sick tax for the hospital and we are providing an
additional $17 million.

In 1999, State employees in the Tallahassee and Big Bend Area raised
$1,975,000!
This year, the Florida legislature increased funding by over $1 billion.
```

REGEXPER

You thought you only had two problems...

- Changelog
- Documentation
- Source on GitHub

```
\$1
```

Display

⬇ Download // 🔗 Permalink

"$1"

Created by Jeff Avallone // Generated images licensed: (cc) BY

So…

What do we need to do to match sentences ending with the word "do" followed by a period?

Changelog
Documentation
</> Source on GitHub

```
do\.$
```

Display

⬇ Download // 🔗 Permalink

○— "do." —[End of line]—○

Character classes with [ ]

A character class matches a single character out of all the possibilities contained in brackets — There are certain rules that apply when specifying these classes that we'll get to in a second

[Tt]hanks

will match these lines from our April 2000 emails

```
Thanks for writing.
Thanks for writing.
Thanks Gary.
Thanks for writing and have a wonderful evening.
Thanks you Shelley.
thanks Lourdes.
Thanks Mary.
thanks for writing.
thanks so much for your email.
thanks Lynette.
Thanks Fonda for your email.
Thanks Dr. Gordon.
Thanks Jennifer.
thanks Bob.
Thanks for writing.
thanks Ed.
thanks Diane for your comments.
Thanks for your suggestion in this regard.
```

[Tt]hanks

Display

⬇ Download   //   🔗 Permalink

One of:
"T"
"t"
"hanks"
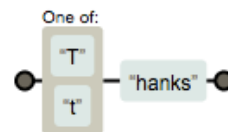
Character classes with [ ]

In terms of the rules that work within character classes, you can specify a range of letters [a-z] or [A-Z] or numbers [0-9] — Keep in mind that the order within the character class doesn't matter, it specifies a bag of characters from which we select one item

[0-9] years

will match these lines from our April 2000 emails

```
Don, the economic development road fund has been in business for over 20 years.

I started playing in earnest about 12 years ago.

Over the last two years, we have increased funding to our child welfare system by
50% which is a greater increase than the previous 8 years.

86.8% of Florida inmates over 65 years of age are in prison for committing a
violent offense.

But Ms. Honan, spending 18 years on death row is not cautious.
```
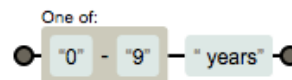
REGEXPER

You thought you only had two problems...

▦ Changelog
🖺 Documentation
</> Source on GitHub

[0-9] years

Display

⬇ Download // 🔗 Permalink

One of:
● "0" - "9" — " years " ●

Created by Jeff Avallone // Generated images licensed: (cc) BY

Character classes with [ ]

When used at the beginning of a character class "^" is also a metacharacter and it indicates matching characters NOT in the indicated class

[^?.]$

will match these lines from our April 2000 emails

```
Wow!

However, I worry about you writing an email at 4:00 am!

Valencia Community College is leading the way in developing business partnerships!

What a week!!!

:) >

I doubt the Cuban Government is paying!

yes
```

More metacharacters: |

This translates to "or" — We can use it to combine expressions, the subexpressions being called alternatives

remember|forget

will match these lines from our April 2000 emails

```
I don't remember this being a letter sent to me at the beginning.

I don't remember this one.

forget that I wasn't notified.
```

More metacharacters: |

This translates to "or" — We can use it to combine expressions, the subexpressions being called alternatives

year|month|day

will match these lines from our April 2000 emails

```
It was a sad day for our country and for our state yesterday.

We have more than quadrupled funding in two years.

Some progress has been made this year but much of it is federal.

The armed extraction of Elian yesterday was inappropriate and unnecessary.

They are given two years to transition to english.

I agree that there must be means by which we strive to heal the wounds opened in the last
few months.

Saturday morning is always good.

I hope you are having a good day.
```

More metacharacters: |

The alternatives an be real expressions and not just literals

^[Kk]atelemail\.$

will match these lines from our April 2000 emails

```
Thanks for your email.

kate can you get terry mcGarrity to respond.

thanks Mr. Cowherd for your email.

kate can you get terry mcGarrity to respond.

You don't know how much I appreciate getting your email.

Kate, this women seems like a very nice person.

Thanks Fonda for your email.

Kate, where do we stand on this?

This is another well written and thought provoking email.
```

regexper.com/#%5E%5BKk%5Date%7Cemail%5C.%24

REGEXPER

You thought you only had two problems...

Changelog
Documentation
</> Source on GitHub

`^[Kk]ate|email\.$`

Display

⬇ Download // 🔗 Permalink

One of:
"K"
"k"
Start of line — "ate"
"email." — End of line
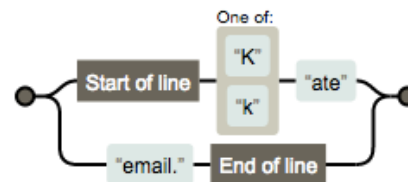
Created by Jeff Avallone // Generated images licensed: (cc) BY

More metacharacters: (and)

Subexpressions are often contained in parentheses to constrain the alternatives in some say

^(I hope‖ promise)

Later we will see that we can identify each subexpression separately, allowing us to extract the content they match

More metacharacters: ?

The question mark indicates that the indicated expression is optional

George(W\.)? Bush

will match references to "George W. Bush" or just "George Bush"

regexper.com/#George(%20W%5C.)%3F%20Bush

REGEXPER

You thought you only had two problems…

```
George( W\.)? Bush
```

Display

⬇ Download // 🔗 Permalink

Created by Jeff Avallone // Generated images licensed: (cc) BY

More metacharacters: * and +

The * and + signs are metacharacters used to indicate repetition — The * means "any number, including zero, of the item" and + means "at least one of the item"

\(.*\)

will match these lines in Jeb Bush's emails from April of 2000

```
It works well for education (what percentage increase?)

I disagree with what you said (on reflection) when we talked
today that the law enforcement and firefighters won't get
engaged in the current situation.

The local communities should step up to the plate (most have
and are not worried about continuing to do so) to make this
work.

On a subject as sensitive as partial birth abortion (where my
views have been clear for years) to get it wrong I think
deserves clarification.

let us hope the session (and reorg) are almost over!
```

REGEXPER

You thought you only had two problems…

Changelog
Documentation
Source on GitHub
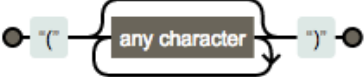
```
\(.*\)
```

Display

Download // Permalink

"(" any character ")"

More metacharacters: * and +

The * and + signs are metacharacters used to indicate repetition — The * means "any number, including zero, of the item" and + means "at least one of the item"

[0-9]+-[0-9]+-[0-9]+

will match these lines in Jeb Bush's Inbox from Jan-May of 2000

```
Phone: 407-240-1891

In reference to your letter dated october 29, 1998 in which you offer to
help me with my inmigration question, i am a us citizen who is petition for
my husband (a mexican citizen)  petition #SRC-98-204-50114 his name is
FRANCISCO JAVIER CORTEZ HERNANDEZ.

Fax: 407-888-2445

Pager:    850-301-8072

Cell: 407-484-8167

The Reverned uses his pager# 813-303-4726 to get in contact with, or you may email
and I will get in touch with him.
```
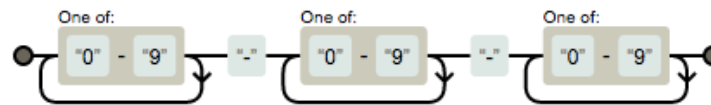
Changelog
Documentation
Source on GitHub

[0-9]+-[0-9]+-[0-9]+

Display

Download // Permalink

A machine view again...

Recall that we can think of a regular expression as a kind of machine that moves along a string, examining each character; we can now revisit that
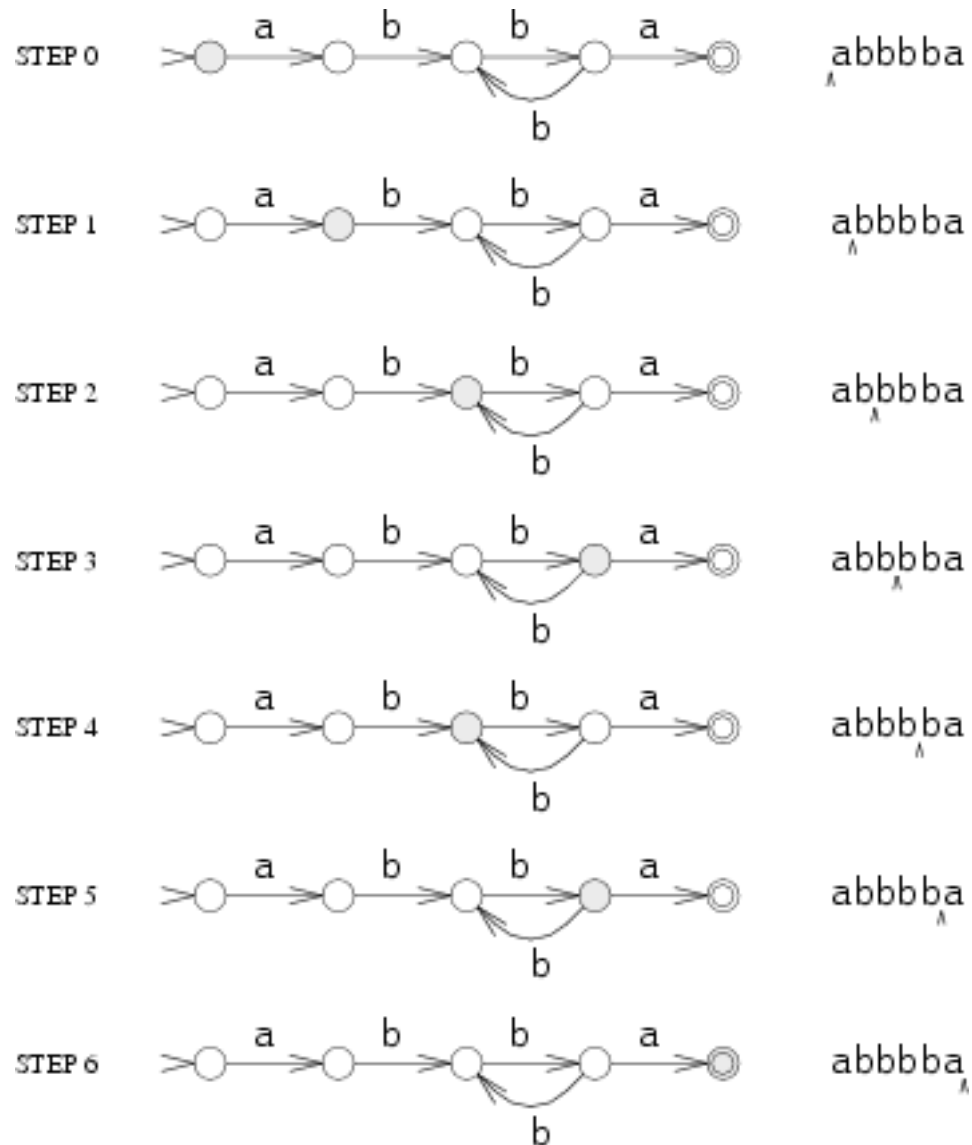
Suppose we have the simple pattern a(bb)+a, which means matching the the character a, followed by some number of double b's, followed by a final a

The state machine that would do this is given graphically as
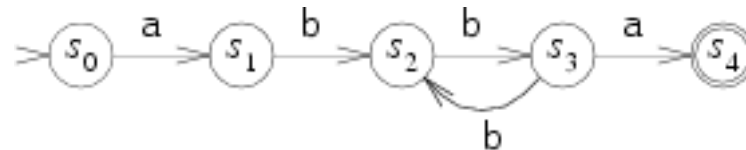
A machine view again...

And here is how it would process the string abbbba

STEP 0   a   b   b   a   b   abbbba

STEP 1   a   b   b   a   b   abbbba

STEP 2   a   b   b   a   b   abbbba

STEP 3   a   b   b   a   b   abbbba

STEP 4   a   b   b   a   b   abbbba

STEP 5   a   b   b   a   b   abbbba

STEP 6   a   b   b   a   b   abbbba

More metacharacters: { and }

{ and } are referred to as interval quantifiers — they let us specify the minimum and maximum number of matches of an expression

I (\w+){1,7}your

will match these lines in Jeb Bush's emails from April of 2000

```
I will check out your concerns.

I appreciate your writing.

I cannot answer your very good questions.

I appreciate your support of One Florida.

I appreciate your kind remarks.

I appreciate your writing.

I hope you and yours have had a great Easter.

They are, in their great majority, god fearing, patriotic Americans who
I would imagine share your values.

Again, I appreciate your writing and I hope you have a joyous Easter.

While I respectfully disagree with your position, I do hope you and
your family have a joyous Easter.
```
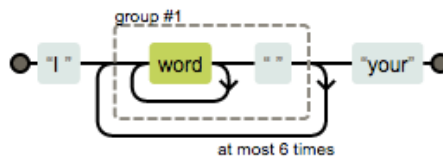
regexper.com/#I%20(%5Cw%2B%20)%7B1%2C7%7Dyour

REGEXPER

You thought you only had two problems…

Changelog
Documentation
</> Source on GitHub

```
I (\w+ ){1,7}your
```

Display

⬇ Download // 🔗 Permalink

group #1

"I " word " " "your"

at most 6 times

Created by Jeff Avallone // Generated images licensed: (cc) BY

More metacharacters: { and }

{m,n} means at least m but not more than n matches

{m}   means exactly m matches

{m,}   means at least m matches

And so…

How would we skim these emails for specific kinds of numbers? Credit card numbers? Social Security numbers?

More metacharacters: (and) revisited

In most implementations of regular expressions, the parentheses not only limit the scope of alternatives divided by a "|", but also can be used to "remember" text matched by the subexpression enclosed

We refer to the matched text with \1, \2, etc.

More metacharacters: (and) revisited

… so the expression

' ([a-zA-Z]+) \1 '

will match these lines in Jeb Bush's inbox from January of 2000

```
I feel this is a win win situation for the Governor, the Reverend and
the people that need help.

I insisted that that be the outcome in that court and that we did not
recede from that position.

I guess you're embarrassed that that line got out.
```
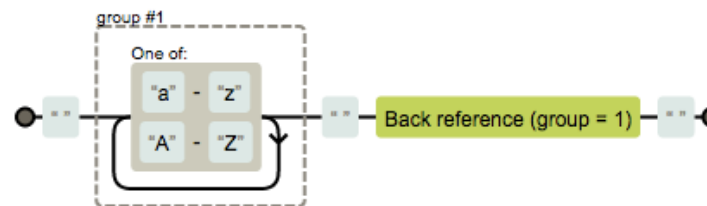
regexper.com/#%20(%5Ba-zA-Z%5D%2B)%20%5C1%20

REGEXPER

You thought you only had two problems...

Changelog
Documentation
</> Source on GitHub

```
([a-zA-Z]+) \1
```

Display

⬇ Download // 🔗 Permalink

Created by Jeff Avallone // Generated images licensed: (cc) BY

Resources

The presentation here is meant to give you a flavor of how regular expressions are structured; you have seen the major metacharacters and to use them to create patterns

On our course home page we provide some good descriptions of regular expressions; in addition, the site

```
http://www.regular-expressions.info/
```

is an excellent resource

Most Visited ▾   Getting Started   Latest Headlines ⟩⟩

[.*] txt2re: headache relief for progra...   +

txt2re

headache relief for programmers :: regular expression generator   (perl php python java javascript

Enter the string that you want to use a regular expression on:

1

[19/Aug/2010:00:00:02 −0700]

(Show Matches)

Select the elements that you want to extract to run regular expression generator:

2

[	1	9	/	A	u	g	/	2	0	1	0	:	0	0	:	0	0	:	0
c [	c 1	c 9	c /	c A	c u	c g	c /	c 2	c 0	c 1	c 0	c :	c 0	c 0	c :	c 0	c 0	c :	c 0

d 1 d 9   W A W u W g   d 2 d 0 d 1 d 0   d 0 d 0   d 0 d 0   d 0 d

day 19   var Aug   day 10   day 00   day 00   day

int 19   word Aug   year 2010   int 00   int 00   int

month Aug   int 2010

unixpath /Aug/2010

ddmmmyyyy 19/Aug/2010

sbraces

What to click?   extract any integer in this position ◀── (int)━━━━━━(2006)

extract this integer (2006) in this position ◀── (int)━━━━━━(2006)

Done

SUBSCRIBE 🔊

TRE

The free and portable approximate regex matching library.

About

WP Greet Box icon   Hello there! If you are new here, you might want to **subscribe to the RSS feed** for updates on TRE.   ✗

Powered by WP Greet Box

TRE is a lightweight, robust, and efficient POSIX compliant regexp matching library with some exciting features such as approximate (fuzzy) matching.

The matching algorithm used in TRE uses linear worst-case time in the length of the text being searched, and quadratic worst-case time in the length of the used regular expression. In other words, the time complexity of the algorithm is $O(M^2N)$, where M is the length of the regular expression and N is the length of the text. The used space is also quadratic on the length of the regex, but does not depend on the searched string. This quadratic behaviour occurs only on pathological cases which are probably very rare in practice.

Features

TRE is not just yet another regexp matcher. TRE has some features which are not there in most free POSIX compatible implementations. Most of these features are not present in non-free implementations either, for that matter.

LINKS

Hacker Boss blog

Perl Compatible Regular Expressions

RegexBuddy

Stack Overflow regex tag