Regulární výrazy prakticky

Martin Bruchanov — bruxy@regnet.cz

4. října 2013

Co, jak a proč?

„*Regulární výraz* je vzor, který udává množinu řetězců znaků; říká jak mají vypadat odpovídající řetězce.“ — Ken Thompson, autor editoru QED

- Možná znáte:
  - Vzory v názvech souborů: `*.*`, `*.jpg`, `???.*`, `...`
  - Vzory v SQL dotazech:

    ```
    SELECT * FROM tabulka WHERE name LIKE 'a%' and 'z_';
    ```
  - Popis syntaxe programovacích jazyků Backusova–Naurova Forma (BNF): `n ::= {_|A..Z|a..z}`, `d ::= {0..9}`, `id ::= n,{n,d}`
- Čím se budeme zabývat:
  - Viz `man 7 regex` – regulární výrazy POSIX (regex, RE)
  - základní (BRE) / rozšířené (ERE)
  - `grep, sed, awk, vi, vim, bash`
  - `perl, python`

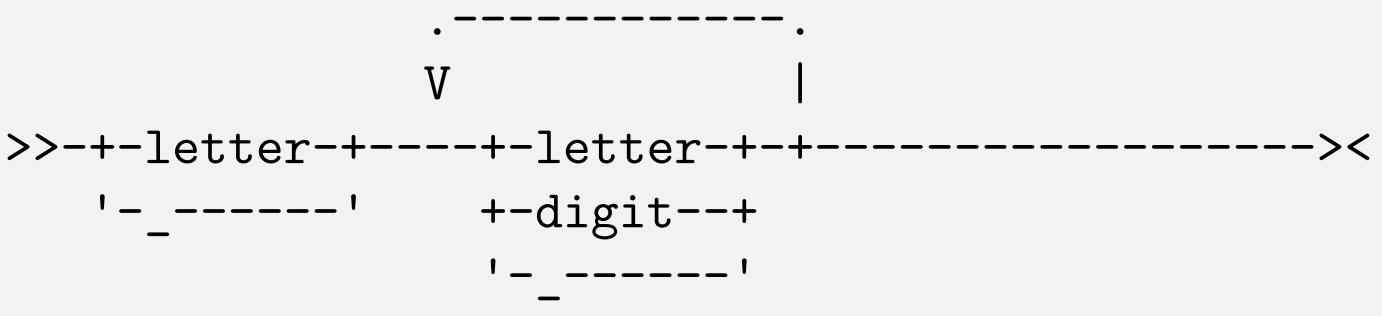
Metaznaky

- `.` (tečka) – libovolný znak (kromě „\n“)
- `[...]`/`[^...]` – množina/negovaná množina znaků
- `^`/`$` – začátek/konec řádku nebo řetězce
- `\(...\)`/`(...)` – seskupení výrazu (atom) BRE/ERE
- `\|`, `|` – nebo, alternativní výrazy (ERE)
- `\n` – n-tý zachycený podvýraz
- Další častá rozšíření:
  - `&` – poslední nahrazený řetězec (sed, ViM)
  - `\<` / `\>` – začátek/konec slova

Příklad 1.: Test názvu proměnné (indentifikátoru) v jazyce C.

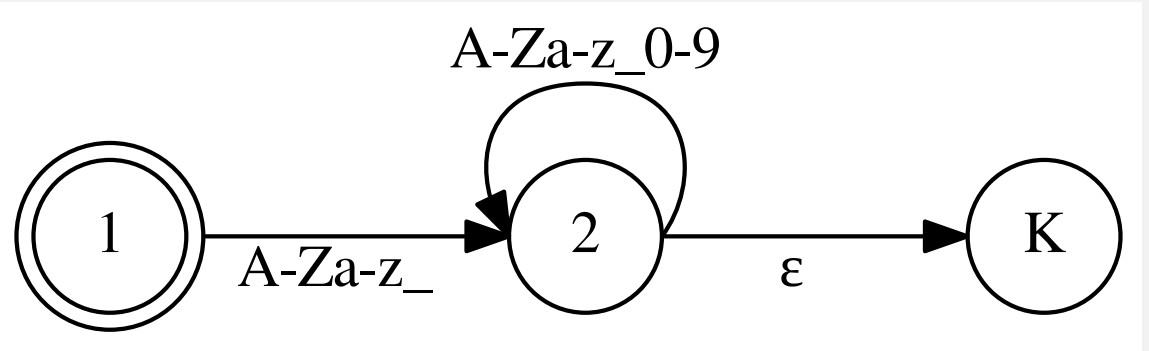
Název proměnné v jazyce C může obsahovat malé a velké znaky anglické abecedy, podtržítko a číslo. Název proměnné nesmí začínat číslem.
Z referenční příručky jazyka C:

```
nondigit ::= {_|A..Z|a..z}
digit    ::= {0..9}
identifier ::= nondigit,{nondigit|digit}
```



Příklad 1.: Regulární výraz

- `^[A-Za-z_][A-Za-z_0-9]*$`
- `^[[:alpha:]_][[:word:]]*$`



Kvantifikátory

Stand.	Líný	Rozsah
`*`	`*?`	0 a víc
`+`	`+?`	1 a víc (ERE)
`?`	`??`	0 nebo 1 položka (ERE)
`{n}`	`{n}?`	přesně $n\times$
`{n,}`	`{n,}?`	minimálně $n\times$
`{n,m}`	`{n,m}?`	min. $n\times$ a max. $m\times$.

Příklady:

- `a*b` – aabc abc bc
- `a?b` – aabc abc bc
- `a+b` – aabc abc bc
- `[els]{1,3}` – all in the darkness
- `Pe(t|p)a` – Pera Peta Pepa
- `".*"` – "Ahoj" "Hello"
- `"[^"]*"` – "Ahoj" "Hello"

- Pozor na rozdíl Basic-RE: `\{m,n\}`, Extended-RE: `{m,n}`
- Líná varianta pochází z Perlu.

Množiny

Meta	POSIX	Unicode	Množina	Popis
`\d`	`[:digit:]`	`\p{IsDigit}`	`[0-9]`	Číslice
`\D`		`\P{IsDigit}`	`[^0-9]`	Cokoliv mimo číslici
`\s`	`[:space:]`	`\p{IsSpace}`	`[ \t\n\r\f]`	Bílý znak
`\S`		`\P{IsSpace}`	`[^ \t\n\r\f]`	Cokoliv mimo bílého znaku
`\w`	`[:word:]`	`\p{IsWord}`	`[a-zA-Z0-9_]`	Znaky identifikátorů
`\W`		`\P{IsWord}`	`[^a-zA-Z0-9_]`	Cokoliv mimo znaků identifikátorů
`\b`				hranice slova, opakem je `\B`
	`[:alnum:]`	`\p{PosixAlnum}`	`[A-Za-z0-9]`	Alfanumerické znaky
	`[:xdigit:]`	`\p{PosixXDigit}`	`[A-Fa-f0-9]`	Hexadecimální čísla
	`[:print:]`	`\p{PosixPrint}`	`[\x20-\x7E]`	Tisknutelné znaky
	`[:alpha:]`	`\p{PosixAlnum}`	`[A-Za-z]`	Abecední znaky

Porovnání nástrojů

	grep/sed	grep -E	awk	vim	Perl
znaky					
libovolný znak (mimo \n)	`.`	`.`	`.`	`.`	`.`
množina znaků	`[ ]`	`[ ]`	`[ ]`	`[ ]`	`[ ]`
kromě těchto znaků	`[^ ]`	`[^ ]`	`[^ ]`	`[^ ]`	`[^ ]`
opakování					
libovolný počet	`*`	`*`	`*`	`*`	`* *?`
alespoň jeden	`\+`	`+`	`\+`	`\+`	`+ +?`
nanejvýš jeden	`\?`	`?`	`?`	`\?`	`? ??`
přesně $n\times$	`\{n\}`	`{n}`	`{n}`	`\{n\}`	`{n}`
minimálně	`\{min,\}`	`{min,}`	`{min,}`	`\{min,\}`	`{min,}`
min až max	`\{min,max\}`	`{min,max}`	`{min,max}`	`\{min,max\}`	`{min,max}`
pozice					
začátek řádku	`^`	`^`	`^`	`^`	`^`
konec řádku	`$`	`$`	`$`	`$`	`$`
začátek slova	`\<`	`\<`	`\< \y`	`\<`	`\A \b`
konec slova	`\>`	`\>`	`\> \y`	`\>`	`\z \b`
závorky a paměť					
skup. se zapamatováním	`\( \)`	`( )`		`\( \)`	`( )`
skupina bez paměti			`( )`		`(?: )`
třetí zapamatovaný	`\3`	`\3`	`\3`	`\3`	`$3`

Neinteraktivní proudový editor sed

- K čemu a jak to funguje? (Podpora UTF-8.)
- `sed [-n] -e '`*`příkaz`*`[;`*`příkaz`*$_2$`]' [-f `*`skript`*`] `*`vstup(y)`*
- Nahrazování (s):
  `sed s/vzor/náhrada/flag < vstup.txt > vystup.txt`
  Flagy: `g` – vše pokryté regexem, n – pořadí výskytu v řádku
- Rozsah: `'1,10 `*`příkaz`*`'`, do konce `'10,$ `*`příkaz`*`'`
- Vypsání části (p): `sed -n '5,$p' vstup.txt`
- Vyhledávání: `sed -n '/regex/p' vstup.txt`
- Náhrada znaků (y): `y/ěščřžýáíé/escrzyaie/`
- Vymazání řádků (d): `'/^[[:blank:]]*#/d'`, `'/^$/d'`
- Vlož před řádek (i): `'1i\text'` před první řádek vlož „text“
- Vlož za řádek (a): `'$a\konec'` za poslední vlož „konec“

GNU Bourne-Again SHell

```
1 #!/bin/bash
2 name="LinuxDays2013"
3 if [[ $name =~ L[a-zA-Z]*([0-9]+) ]]
4 then
5     echo ${BASH_REMATCH[1]}
6 fi
```

- Podpora od verze 3.x (`$BASH_VERSION`)
- `$BASH_REMATCH`, `${BASH_REMATCH[0]}` – celý řetězec
- `${BASH_REMATCH[1]}` – podskupina „jedno a více čísel“

Python

```
1 #!/usr/bin/env python
2 import re
3 name = "LinuxDays2013"
4
5 match = re.match(r'L[a-zA-Z]*([0-9]+)', name)
6 if match:
7     print match.group(1)
```

- Podpora perlovského stylu regexů
- Použijte surové řetězce `r'`*regex*`'` (problém escape-sekvence)
- `object = re.match(regex, vstup, flagy)`

Perl

```
1 #!/usr/bin/env perl
2
3 $name = "LinuxDays2013";
4
5 if ( $name =~ m/L[a-zA-Z]*([0-9]+)/ ) {
6     print $1 . "\n"
7 }
```

- Zpětná reference se provádí pomocí $\$n$
- Operátor `m/regex/flagy` vrací true při nalezení
- Nahrazení `$name =~ s/Linux/Windows/;`