

+ 10

Para saber mais: POSIX vs Metacaracteres



18%

ATIVIDADES
10 DE 13DISCORD
ALURAFÓRUM DO
CURSOVOLTAR
PARA
DASHBOARDCONHEÇA O
VITRINE.DEVMODO
NOTURNO

67k xp



Uma abordagem comum para criar padrões de expressões regulares é usar **metacaracteres**, que são caracteres com significados especiais dentro de uma Regex. **Mas você sabia que é possível trabalhar com expressões regulares além dos meta-chars?**

Outra abordagem para a construção de padrões é com o uso de POSIX (Portable Operating System Interface for Unix, Interface de Sistema Operacional Portátil para Unix). O POSIX é uma padronização que define algumas funcionalidades suportadas por sistemas operacionais baseados em UNIX e garante a comunicação universal entre os sistemas.

E como funciona o uso do POSIX?

O POSIX chama uma expressão entre colchetes (em inglês, “bracket expression”), que é diferente dos usos de metacaracteres. Por exemplo, conseguimos capturar as ocorrências de todos os números em um arquivo baseado em texto com o comando:

```
grep '[:digit:]' caminho_do_arqui
```

COPIAR CÓDIGO



18%

ATIVIDADES
10 DE 13DISCORD
ALURAFÓRUM DO
CURSOVOLTAR
PARA
DASHBOARDCONHEÇA O
VITRINE.DEV

67k xp



Vamos entender melhor com o nosso projeto?

Em nosso **database.csv** , podemos capturar todos os números com o comando `grep` `'[[:digit:]]'` /home/camila/database.csv , como na imagem a seguir:

```
camila@DESKTOP-49MSR11:~$ grep '[[:digit:]]' /home/camila/database.csv
Rogério Marco Bezerra Filho,geraldo@email.com,(49) 92361-2598,83885126486,22/07/1995
Clebson Kauê Assunção Sobrinho,clebinho@email.com.br,(95) 92589-4243,18440126387,19/03/1959
Francisco Chico de Dias,chico@email.com.br,(67) 92846-3472,88544915143,17/10/2004
Bóris Thiago Carrara Mendes,borio@email.com,(85) 92657-6956,48275258979,17/06/1999
Benedito Nivaldo de Cruz,benet@email.com,(44) 92891-2952,87769158168,15/12/1997
Anna Dara Brito,aninha@email.com,(64) 92967-8938,71642300721,04/03/1994
Suzana Galindo de Quintana,suzanaquintana@email.com,(28) 93481-2521,84658458594,19/11/1964
Jason Raul Burgos Leal de Azevedo,jason@email.com.br,(88) 93602-3624,826.580.743-01,13 05 1973
Júlieta Correia Nhrin,juliet@gmail.com,(13) 92977-7223,752.873.488-19,22111996
Berenice Sabrina Bezerra de Maldonado,berenice@email.com,(28) 3547-5161,764.682.144-67,28.03.1984
camila@DESKTOP-49MSR11:~$
```

Identificamos na imagem que o retorno são todas as ocorrências de dígitos, assim os números aparecem destacados na cor vermelha. A partir disso, notamos que a sintaxe de expressão entre colchetes acionada pelo POSIX é similar a classe de caracteres utilizada com JavaScript e o *shorthand* `\d` , como podemos confirmar no código a seguir:

```
const textoComNumeros = "Números: ,
const patternNumeros = /\d+/g;
const resultados = textoComNumeros.
```

```
console.log(resultados)//["24","927
```

[COPIAR CÓDIGO](#)

18%

ATIVIDADES
10 DE 13DISCORD
ALURAFÓRUM DO
CURSOVOLTAR
PARA
DASHBOARDCONHEÇA O
VITRINE.DEV

- Primeiro, criamos uma variável em texto que contém letras e números, a `const textoComNumeros ;`
- Em seguida a `patternNumeros` armazena uma regex para corresponder a um ou mais dígitos;
- Por fim, na `const = resultados` usamos o método `match()` para encontrarmos a ocorrência e imprimimos o resultados no console.

Notamos que o resultado são todas as sequências de números e as letras não aparecem no console.

Mas qual podemos utilizar então?

Ambos os procedimentos têm suas vantagens e desvantagens, o importante é entender qual o contexto e o problema que desejamos solucionar para escolher a melhor opção. Por conseguinte, também precisamos conhecer as expressões POSIX e os meta-chars, confira as tabelas abaixo:



67k xp



POSIX



18%

ATIVIDADES
10 DE 13DISCORD
ALURAFÓRUM DO
CURSOVOLTAR
PARA
DASHBOARDCONHEÇA O
VITRINE.DEV**Bracket Expression****Significado**

<code>[[:digit:]]</code>	Qualquer dígito.
<code>[[:alpha:]]</code>	Qualquer caractere alfabético.
<code>[[:alnum:]]</code>	Qualquer caractere alfanumérico.
<code>[[:blank:]]</code>	Espaço em branco ou caractere de tabulação.
<code>[[:space:]]</code>	Qualquer caractere de espaço em branco.
<code>[[:lower:]]</code>	Qualquer letra minúscula.
<code>[[:upper:]]</code>	Qualquer letra maiúscula.
<code>[[:print:]]</code>	Qualquer caractere imprimível, incluindo espaço em branco.
<code>[[:punct:]]</code>	Qualquer caractere de pontuação.
<code>[[:graph:]]</code>	Qualquer caractere imprimível, exceto espaço em branco.
<code>[[:xdigit:]]</code>	Qualquer dígito hexadecimal (0-9, A-F, a-f).
<code>[[:cntrl:]]</code>	Qualquer caractere de controle.

As expressões entre colchetes, conhecidas como "bracket expressions," são utilizadas nas expressões regulares POSIX para corresponder a categorias específicas de caracteres. Elas simplificam a criação de padrões de correspondência personalizados com base em categorias de caracteres, em vez de caracteres individuais.

Metacaracteres em Regex



67k xp





18%

ATIVIDADES
10 DE 13DISCORD
ALURAFÓRUM DO
CURSOVOLTAR
PARA
DASHBOARDCONHEÇA O
VITRINE.DEV

Metacaractere

.	Qualquer caractere, exceto quebras de
*	Zero ou mais ocorrências do caractere
+	Uma ou mais ocorrências do caractere
?	Zero ou uma ocorrência do caractere o
	Alternância, corresponde a um dos pac
()	Grupo de captura, agrupa caracteres pa
[]	Classe de caracteres, corresponde a qu
[^]	Classe de caracteres negada, correspon
^	ncora de início de linha, corresponde a
\$	ncora de final de linha, corresponde ac
\	Escape, permite escapar metacaractere
{ }	Quantificador personalizado, especifica

Esses são alguns dos metacaracteres mais comuns usados em expressões regulares para criar padrões de correspondência flexíveis e poderosos em texto.

Para conhecer mais sobre o POSIX e Expressões Regulares você pode acessar o artigo [POSIX Basic Regular Expressions da RegexBuddy](https://www.regular-expressions.info/posix.html) (<https://www.regular-expressions.info/posix.html>).



67k xp

