10

Para saber mais: POSIX vs Metacaracteres

18%

ATIVIDADES
10 DE 13

DISCORD
ALURA

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD

CONHEÇA O
VITRINE.DEV

MODO
NOTURNO

Uma abordagem comum para criar padrões de 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:

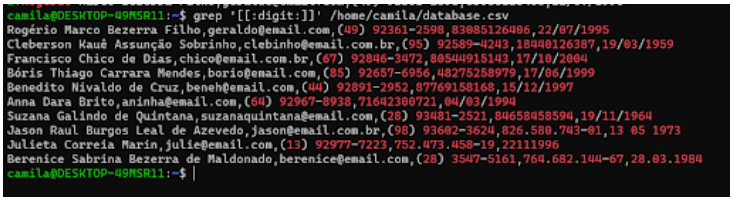
```
grep ‘[[:digit:]]’ caminho_do_arqui
```

67k xp

COPIAR CÓDIGO

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:



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

- 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:

POSIX

Bracket Expression	Significado
`[[:digit:]]`	Qualquer dígito.
`[[:alpha:]]`	Qualquer caractere alfabético.
`[[:alnum:]]`	Qualquer caractere alfanumérico.
`[[:blank:]]`	Espaço em branco ou caractere de
`[[:space:]]`	Qualquer caractere de espaço em
`[[:lower:]]`	Qualquer letra minúscula.
`[[:upper:]]`	Qualquer letra maiúscula.
`[[:print:]]`	Qualquer caractere imprimível, in
`[[:punct:]]`	Qualquer caractere de pontuação.
`[[:graph:]]`	Qualquer caractere imprimível, ex
`[[:xdigit:]]`	Qualquer dígito hexadecimal (0-9,
`[[:cntrl:]]`	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

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 pad
()	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).