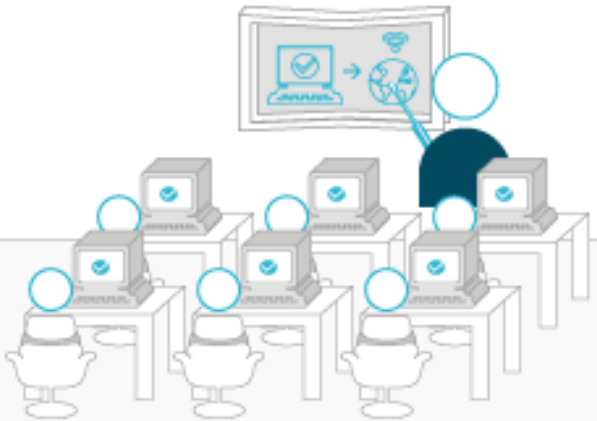
Título do Curso:

Administração de Sistemas GNU/Linux

Conteúdo do curso

ADMINISTRAÇÃO DE SISTEMAS GNU/LINUX

- Tópico 1: Introdução ao sistema operacional GNU/Linux.
- Tópico 2: Introdução ao Shell e comandos básicos.
- Tópico 3: Manipulação de conteúdos com comandos no Shell.
- Tópico 4: Comandos para gerenciamento do sistema e do Hardware.
- Tópico 5: Editor de Texto VI.
- Tópico 6: Administração de usuários e grupos.
- Tópico 7: Gerenciamento de permissões.
- Tópico 8: Gerenciamento de processos.
- Tópico 9: Sistemas de arquivos e particionamento.
- **Tópico 10: Expressões regulares.**
- Tópico 11: Introdução ao Shell Script.
- Tópico 12: Gerenciamento de Pacotes.
- Tópico 13: Agendamento de tarefas (cron) e Backup.



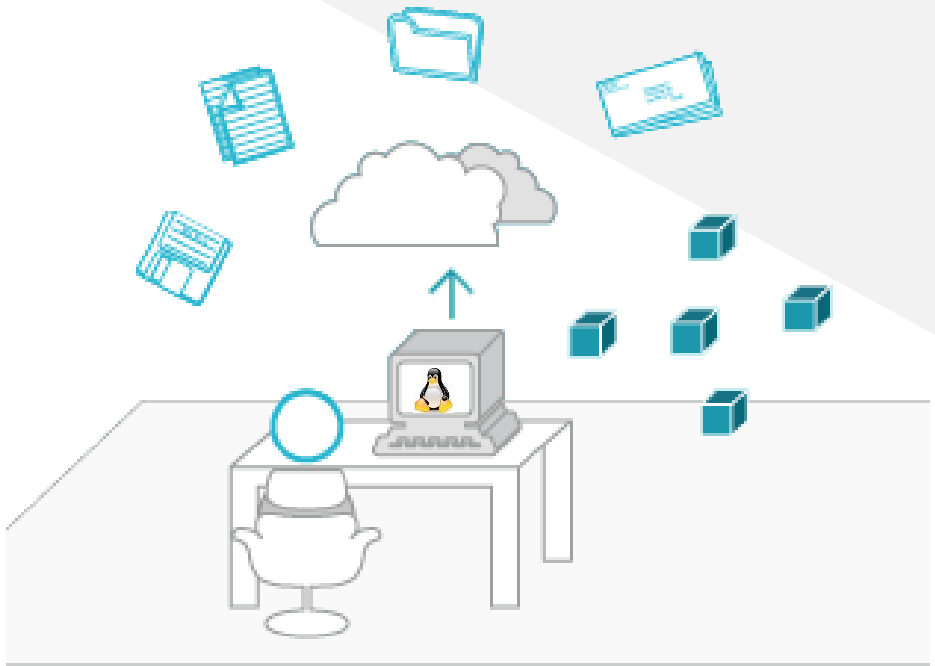
Tópico 10

Expressões regulares

Expressões Regulares

REGEX

- Neste slide teremos conceitos e fundamentos importantes sobre o funcionamento e o uso de expressões regulares:
  - Conceitos e aplicabilidade;
  - Metacaracteres e suas funções.



Conceitos e aplicabilidade

Tópico 10: Expressões regulares.

Introdução a Expressões Regulares – HISTÓRIA!!!

- Antes mesmo da existência de sistemas operacionais e programas de computador, tivemos o “embrião” sobre o tema.
- Em 1943, dois neurologistas (os “pais” da “ER”) publicaram um estudo que teorizava o funcionamento de nossos neurônios.
- Alguns anos depois, um matemático (o “parteiro” da “ER”) descreveu os modelos deste estudo na forma algébrica, utilizando símbolos para representar o que fora chamado de “*regular sets*”.
- A “Expressão Regular” surgiu a partir desta notação simbólica, que foi estudada por aproximadamente 20 anos pelos matemáticos da época.

- Apenas em 1968 a ER foi introduzida na computação, através de um algoritmo de busca no editor “qed” (que posteriormente deu origem ao “ed”).

Introdução a Expressões Regulares

- No editor de texto “ed” havia um comando de contexto “g” com suporte a expressão regular, que posteriormente dependia do comando “p” (print), desta forma sua sintaxe era “g/RE/p” (*Global Regular Expression Print*), dando origem ao comando “grep” e derivados.

- Com o tempo diversos aplicativos passaram a utilizar ER, como o “sed”, “awk”, “vi”, “find”, entre outros. Cada um com suas particularidades.

- Apesar do tema ser antigo, o conteúdo é o mesmo ao decorrer destes anos, devido a consistência do tema.
  - OBS.: Um livro ótimo sobre o tema elaborado pelo Aurélio Marinho Jargas (utilizado como referência desta apresentação) pode ser encontrado em:
    - **http://aurelio.net/regex/guia/**

Introdução a Expressões Regulares

- Afinal, o que são as “Expressões Regulares”???

- Terminologia formal:
  - “É um método de se especificar um padrão de texto”;
  - “São metacaracteres que casam um padrão”;
  - “Uma maneira de procurar um texto que você não lembra exatamente como é, mas tem ideia das variações possíveis”. (JARGAS, “20((0[1689])|(1[26])”).

- Terminologia informal (JARGAS):
  - “Como o alfabeto. Você aprende primeiro as letras individualmente. Depois as sílabas, as palavras, frases e finalmente os textos. Mas no fundo, são apenas letras.”

Introdução a Expressões Regulares

- Agora, vamos a explicação de sua terminologia:
  - “São metacaracteres que casam um padrão”;
    - Metacaracteres → São símbolos e caracteres literais com função específica;
    - Casar → Ato de “coincidir”, bater, equiparar um termo (ou uma string);
    - Padrão → Objetivo da ER, casar/coincidir com um padrão especificado.

- Após descrever a terminologia, vamos aos exemplos rápidos:
  - [rpg]ato = rato, pato, gato;
  - [1-4]5 = 15, 25, 35, 45;
  - n.o = não, nao, neo;
  - M[aeiou]to = Mato, Meto, Mito, Moto, Muto;

Expressões regulares... Servem pra que?

- Temos inúmeras possibilidades em relação a expressões regulares que não seria viável descrever uma lista...

- Podemos dizer que sempre serão úteis quando a busca exigir valores de texto variáveis, como:
  - Data / hora;
  - Endereçamento (IPv4, MAC, IPv6, ...);
  - Dados pessoais (RG, CPF, Endereço, CEP, e-mail, site, login);
  - Campos de texto tabulados ou delimitados por caracteres específicos (como vírgula, ponto e vírgula, dois pontos, etc...);
  - Dados que estão no começo ou no final de uma linha;
  - Entre outras finalidades;

Metacaracteres e suas funções

Tópico 10: Expressões regulares.

Metacaracteres (O que é isso?)

- Segundo AURÉLIO (2006):
  - Cada metacaractere é uma ferramenta que tem uma função específica. Eles servem para dar mais poder às pesquisas, informando padrões e posições impossíveis de se especificar usando somente caracteres normais.
  - Os metacaracteres são pequenos pedacinhos simples que agrupados entre si, ou com caracteres normais, formam algo maior, uma expressão. O importante é compreender bem cada um individualmente, e depois apenas lê–los em sequência.

metacaractere	mnemônico	
.	ponto	Representantes
[]	lista	
[^]	lista negada	
?	opcional	Quantificadores
*	asterisco	
+	Mais	
{}	chaves	

metacaractere	mnemônico	
^	circunflexo	Âncora
$	cifrão	
\b	borda	
\	escape	Outros
|	ou	
()	grupo	
\1	retrovisor	

Metacaracteres (O que é isso?)

- Função e “nome” (mnemônico) de cada metacaractere:

Metacaractere	Mnemônico	Função
.	Ponto	um caractere qualquer
[...]	Lista	lista de caracteres permitidos
[^...]	Lista negada	lista de caracteres proibidos
?	Opcional	zero ou um
*	Asterisco	zero, um ou mais
+	Mais	um ou mais
{n,m}	Chaves	de "n" até "m" (ou seja, intervalo especificado)
^	Circunflexo	início da linha
$	Cifrão	fim da linha
\b	Borda	início ou fim de palavra (limítrofe)
\x	Escape	torna literal o caractere "x" --> Ex.: \$
|	Ou	ou um ou outro
(...)	Grupo	delimita um grupo
\1...\9	Retrovisor	texto casado nos grupos 1...9

Âncoras:
O circunflexo, cifrão e a borda → ^ $ \b

- Os metacaracteres do tipo “âncora” não representam nenhum tipo de caractere para o padrão de busca. Eles definem que o padrão deve estar presente no início ^ ou no final da linha $, bem como delimitá-lo \b.

- Circunflexo (início da linha):
  - ^d → Procura por linhas que começam com a letra “d”;
  - ^[0-9] → Pesquisam por linhas que comecem com um número de 0 a 9;
  - Texto^ → Apenas um “circunflexo após a palavra “texto”;
  - ^^ → Apenas uma linha que começa com o “circunflexo”;
    - OBS.: O metacaractere circunflexo é interpretado como “literal” caso não esteja no início da linha.
- Cifrão (final da linha):
  - fechado.$ → Procura por linhas que terminam com a string “fechado.”;
  - ^$ → Linha vazia.

Âncoras:
O circunflexo, cifrão e a borda → ^ $ \b

- Borda (a limítrofe): Como o nome diz, marca/define uma borda, ou seja, de forma mais específica, delimita uma palavra:
  - Ex.: Vamos supor que desejamos procurar pela palavra “dia” em um texto que possui além da palavra “dia” as palavras “melodia”, “diafragma”, “radial” e “bom-dia!”.
    - dia → dia, melodia, diafragma, radial, bom-dia!;
    - \bdia → dia, diafragma, bom-dia!;
    - dia\b → dia, melodia, bom-dia!;
    - \bdia\b → dia, bom-dia!;
  - OBS.: Entenda que "palavra" aqui é um conceito que engloba [A-Za-z0-9_] apenas, ou seja, letras, números e o sublinhado. Por isso \bdia\b também casa com “bom-dia!” pois o traço e a exclamação não são parte de uma palavra.

Representantes:
O Ponto → .

- Os metacaracteres do tipo representantes são utilizados para especificar/representar caracteres, sendo que cada metacaractere deste tipo casa com apenas um único caractere.

- Ponto (o "curinga" da ER): O ponto pode substituir qualquer caractere, ou seja, casa com qualquer caractere (inclusive, espaço, TAB, @), porém, apenas um caractere na posição em que o mesmo está.
  - n.o = não, nao, neo, n-o;
  - .ala = fala, cala, mala, tala, bala, etc...;
  - <.> = <b>, <i>, <n> (padrão muito utilizado em programação como HTML);
  - 22.15 = 22:15, 22.15, 22 15, 22-15, 22/15, etc...

Representantes: A lista → [.....]

- Lista (a “seletiva”): A lista apenas casa com os caracteres especificados dentro da lista:
  - [fm]ala = fala, mala;
  - n[ãe]o = não, neo;
  - 22[: .]15 = 22:15, 22.15, 22 15;
    - *OPS... E o ponto, não casa com qualquer coisa??*
    - OBS.: Dentro da lista, todo caractere se torna “literal”;
  - Agora vamos supor que desejamos especificar uma hora qualquer... Devemos fazer [012][0123456789]:[012345]..... AAAAAAAAAAAAAA!!!!
  - Através da lista podemos especificar um intervalo qualquer através do traço/hífen.
    - Ex.: [0123456789] = [0-9]

Representantes: Especificar intervalo na lista → [-]

- Lista com intervalos: Através da lista com intervalos, podemos resumir significativamente uma expressão regular.

- Imagine uma lista de parte do alfabeto... Temos no exemplo a seguir a utilização de uma lista com todos os caracteres e outra com o intervalo de caracteres:
  - Ex.: [defghijklmnopqrs] = [d-s]
  - Ex.: [0123456789] = [0-9]

- Desta forma, já podemos representar uma hora qualquer de forma mais inteligente:
  - [012][0-9]:[0-5][0-9]
    - ***E aí? A representação acima está correta?***

Representantes:
A lista negada → [^....]

- Lista negada: da mesma forma que praticamos com o “grep –v”, podemos especificar o contrário da string desejada:
  - [^fm]ala = bala, cala, pala, tala, etc...

- **RESUMO SOBRE LISTAS:**
  - A lista casa com quem ela conhece e tem suas próprias regras.
  - Dentro da lista, todo mundo é normal (literal).
  - Dentro da lista, traço indica intervalo.
  - Um - literal deve ser o último item da lista.
  - Um] literal deve ser o primeiro item da lista.
  - Os intervalos respeitam a tabela ASCII (não use A-z).
  - [:classes POSIX:] incluem acentuação, A-Z não. (Tabela sobre classes POSIX no próximo slide).

[:classes POSIX:] → Não é LISTA!

- Classes POSIX que podem nos salvar em algum momento...

Classe POSIX	Lista similar	Função
[:upper:]	[A-Z]	Letras maiúsculas
[:lower:]	[a-z]	Letras minúsculas
[:alpha:]	[A-Za-z]	Letras maiúsculas e minúsculas
[:alnum:]	[A-Za-z0-9]	Letras e números - Alfanuméricos
[:digit:]	[0-9]	Números
[:xdigit:]	[0-9A-Fa-f]	Números Hexadecimais
[:punct:]	[.,!?:...]	Sinais de Pontuação
[:blank:]	[\t]	Espaço e TAB
[:space:]	[\t\n\r\f\v]	Espaços / Caracteres em branco
[:cntrl:]	-	Caracteres de controle
[:graph:]	[^ \t\n\r\f\v]	Caracteres imprimíveis
[:print:]	[^\t\n\r\f\v]	Imprimíveis e o espaço

Quantificadores: Opcional, asterisco, mais e chaves → ? * + {...}

- Os metacaracteres do tipo quantificadores, servem para indicar o número de repetições para a “entidade” anterior, que pode ser um caractere literal ou um metacaractere;

- Opcional → ? : O opcional é um quantificador que define se a “entidade” anterior aparece 0 ou 1 vez:
  - Ondas? = Onda, Ondas;
  - Fala[r!]? = Fala, Falar, Fala!;

- Asterísco → * : A ocorrência pode aparecer nenhuma, uma ou várias vezes, ou seja, semelhante ao opcional, porém aceita qualquer qtde.:
  - 5*0 = 0, 50, 550, 5550, 555555555555550, ...
  - Fa*la = Fla, Fala, Faala, Faaaaaaaaaaala, ...

Quantificadores: Opcional, asterisco, mais e chaves → ? * + {...}

- Mais → + : semelhante ao asterisco, é utilizado para definir que a “entidade” anterior deve casar pelo menos uma vez, portanto, não é “opcional”:
  - 5+0 = 50, 550, 5550, 555555555555550, ...
  - Fa+la = Fala, Faala, Faaaaaaaaaaala, ...

- Chaves → {....} → (o “controle”): Através das chaves podemos definir o número exato de repetições aceitáveis da “entidade” anterior:
  - 7{1,4} = 7, 77, 777, 7777;
  - egrep ‘^.{27}$’ /etc/passwd = Procura uma linha com exatos 27 caracteres;
  - ^.{10,30}$ = Procura por uma linha que tenha entre 10 e 30 caracteres;

Quantificadores: Opcional, asterisco, mais e chaves → ? * + {...}

- Continuação Chaves → {....} : Através das chaves também podemos omitir a quantidade final, além de especificar um intervalo ou a quantidade exata:
  - {1,3} → de 1 a 3
  - {3,} → pelo menos 3 (3 ou mais)
  - {0,3} → até 3
  - {3} → exatamente 3
  - {1} → exatamente 1
  - {0,1} → zero ou 1 (igual ao opcional)
  - {0,} → zero ou mais (igual ao asterisco)
  - {1,} → um ou mais (igual ao mais)

- Obs.: A chave foi criada após os demais quantificadores (?, *, +);

Outros metacaracteres: O “escape” → \

- Torna um metacaractere em um caractere literal, normal;
  - Imagine se você precisar casar um * como parte do texto?
  - Alguém diz: *“Aaaahhhh!!!! Dentro da lista, todo mundo é literal!”*

- Para não ter que criar uma lista sempre que precisar de um caractere que é um “meta”, mas você quer seu significado literal, podemos utilizar o “escape”, conforme exemplo a seguir:
  - lua[*] = lua* ou lua* = lua*

- Com o “escape”, podemos escapar todos os metacaracteres:
  - \., \[, \], \?, \+, \{, \}, \^ e \$.
- O escape é tão poderoso que pode escapar a si próprio!
  - O \\ casa com uma barra invertida, ou seja, \\ = \ → (caractere literal).

Outros metacaracteres: O curinga (Operador lógico “AND”) → .*

- Também podemos combinar alguns dos metacaracteres, agregando as suas funções. Dentre estas combinações, temos o ponto em conjunto com o asterisco → .*

- Vamos imaginar o seu funcionamento:
  - O ponto representa qualquer caractere...
  - O asterisco representa nenhuma ou qualquer quantidade...
    - Portanto, o que temos? TUDO? NADA?
      - A resposta é AMBOS. Podemos ter qualquer caractere em qualquer quantidade ou simplesmente, NADA.

Outros metacaracteres: O “ou” (Operador lógico “OR”) → |

- Podemos encontrar situações em que mais de uma alternativa ou “string” deve ser procurada. Neste caso podemos utilizar o “ou” → |
  - Boa-tarde|Boa-noite = Boa-tarde, ou, Boa-noite;

- Podem surgir comentários de que sua função é a mesma da “lista” [...], porém, a diferença é que a lista possibilita a substituição de apenas um caractere.

- Geralmente, o operador “OR” é utilizado juntamente com o “grupo”, que veremos no slide a seguir.
  - *(O grupo “multiplica” o poder do “OR”).*

Outros metacaracteres: O Grupo → (...)

- O grupo funciona como uma expressão matemática e permite que sejam inseridos caracteres, metacaracteres e inclusive outros grupos;

- Podemos ampliar o “poder” dos metacaracteres dentro de um grupo.

- Exemplos:
  - (há!)+ → Temos um grupo quantificado pelo “mais”, ou seja, podemos ter há!, há!há!, há!há!há!, e assim sucessivamente.
  - Boa-(tarde|noite) → Boa-tarde, ou, Boa-noite;
  - (in|con)?certo → incerto, concerto, certo;

Outros metacaracteres: O “Retrovisor” → \1 \2 \3 \4 ..... \9

- Ao utilizar um grupo “ganhamos de brinde”... o trecho “casado” pela ER contida no grupo pode ser utilizado posteriormente pelo “retrovisor”;
- Como assim?
- Como o próprio nome já diz, o “retrovisor” tem a função de “olhar para trás” e buscar um trecho já casado em uma ER, que estava contido em um grupo, casando novos trechos contidos na mesma linha:
- Ex.: Vamos imaginar que precisamos procurar por “teco-teco” no texto.
  - Alguém diz: *“Fácil, basta colocar ‘teco-teco’ e pronto...”*
  - Sim, porém, podemos criar uma ER utilizando o retrovisor da seguinte forma:
    - (teco)-\1 = teco-teco

Outros metacaracteres: O “Retrovisor” → \1 \2 \3 \4 ..... \9

- O exemplo do “teco-teco” é apenas didático... Observem estas possibilidades:
  - (lenta)(mente) é \2 \1 lentamente é mente lenta
  - ((band)eira)nte \1 \2a bandeirante bandeira banda
  - in(d)ol(or) é sem \1\2 indolor é sem dor

- Podemos utilizar até 9 retrovisores, ou seja, buscar em uma ER até 9 resultados de grupos que “casaram” com algum texto.

- Observe estes exemplos parecidos com o “teco-teco”, porém, não sabemos qual palavra encontrar...:
  - ([A-Za-z]+)-\1 → quero-quero, teco-teco, tromba-tromba, etc...
  - ([A-Za-z]+)-?\1 → bombom, lili, dudu, zeze, teco-teco, etc...

Atividade Express!!!

- Agora que já sabemos o básico de ER, Vamos praticar:
  - 1 – Crie uma ER que pesquise por uma RG qualquer em um arquivo?
    - [A-Z]{1,2}-[0-9]{1,2}\.[0-9]{3}\.[0-9]{3}
    - [:upper:]{1,2}-[0-9]{1,2}\.[0-9]{3}\.[0-9]{3}
  - 2 – Crie uma ER para verificar todos os arquivos criados/modificados entre às 07:00 e as 22:00?
    - (0[7-9]|1[0-9]|2[0-2]):[0-5][0-9]
  - 3 – Crie uma ER que pesquise pelas possíveis variações dentro de um arquivo texto? (minimercado, supermercado, hipermercado e mercado).
    - (mini|(su|hi)per)?mercado
    - (mini|super|hiper)?mercado
      - OBS.: A resposta está com a fonte na cor branca. Selecione, sapeca um CTRL+C/V no notepad!

Atividade Express!!!

- No início deste slide, fiz uma ER com a citação retirada do livro do Aurélio Marinho Jargas (o cara da ER), conforme abaixo:
  - “Uma maneira de procurar um texto que você não lembra exatamente como é, mas tem ideia das variações possíveis”. (JARGAS, “20((0[1689])|(1[26]))”).

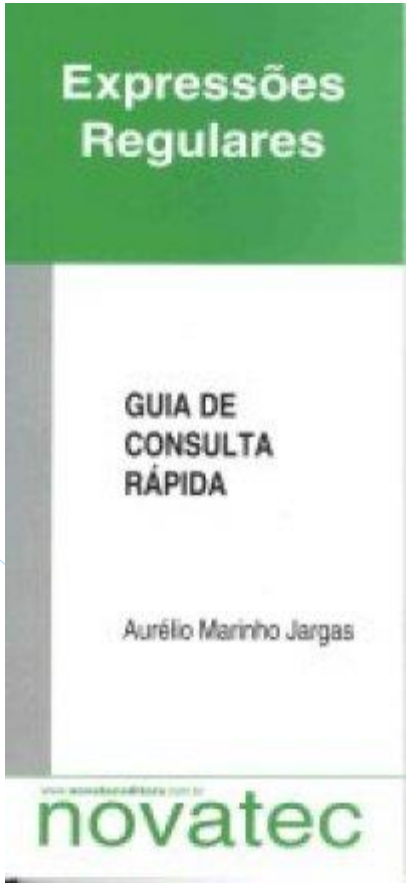
- Agora eu acho que já sabemos identificar as combinações possíveis, certo?
  - Nesta ER temos os anos em que o Jargas publicou seus livros sobre ER:
    - 2001
    - 2006
    - 2008
    - 2009
    - 2012
    - 2016

No próximo slide…

- Tópico 11: Introdução ao Shell Script:
  - Conceitos e aplicabilidade;
  - Criando um Shell Script;
  - Interação com o usuário através do Shell Script.

Referências

- Aurélio Marinho Jargas:
  - http://aurelio.net/regex/guia/



Referências

- BONAN, Adilson Rodrigues. **LINUX – Fundamentos, Prática & Certificação LPI**. Editora: Alta Books. RJ. 2010;

- PEREIRA, Guilherme Rodrigues. **Slides para aula expositiva**. Centro Universitário UNA.

- SILVA, Gleydson Mazioli. **Guia Foca GNU/Linux**. Disponível em: https://guiafoca.org/

DGP

Tecnologia da Informação

Obrigado!

Guilherme Rodrigues