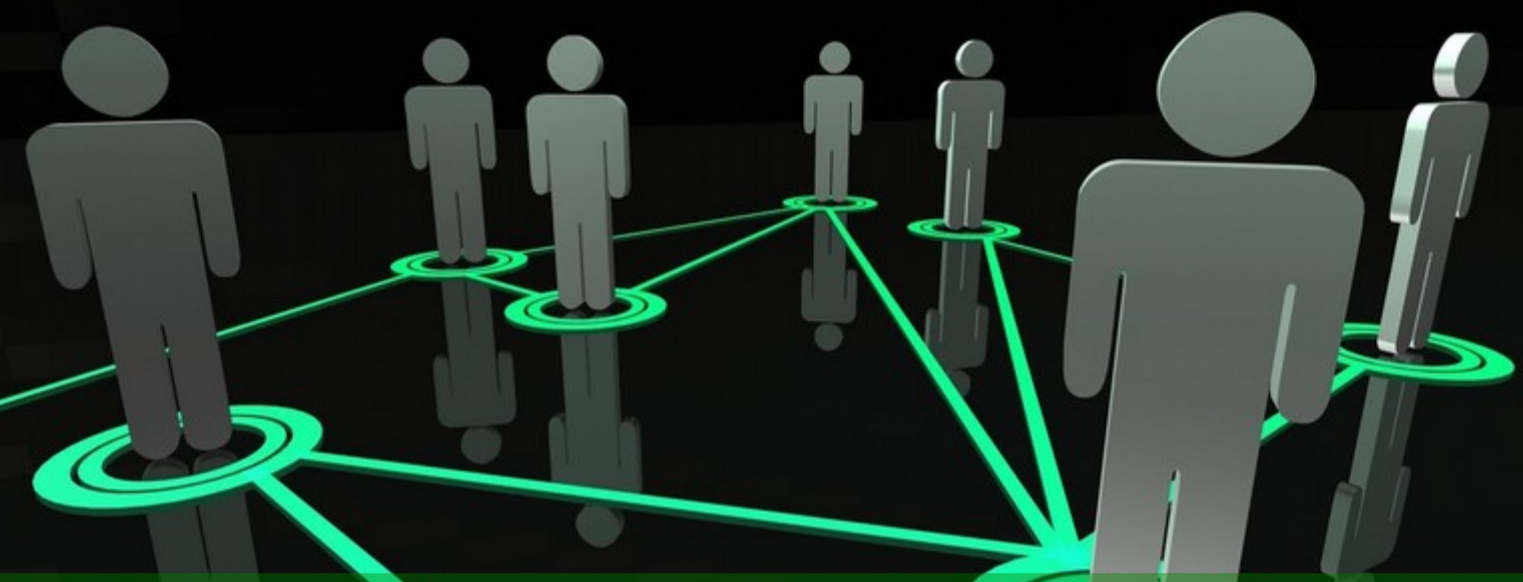


Expressões regulares no BrOffice

nathancolquhoun

Google

Por Julio Neves

Apesar de não ter nada a ver com *Shell*, todos os "shelleiros" usam o BrOffice.

Como vocês já sabem o básico de *Expressões Regulares*, vou inserir esta seção como uma colher de chá - ou será colher de *Shell*?

Obviamente, não mostrarei tudo novamente, mas somente as diferenças que já testei entre as sintaxes das *Expressões Regulares* do *Shell* e do BrOffice, dando ênfase para as surras que tomei para descobrir essas diferenças.

As surras foram maiores, porque à época não se achava documentação sobre o uso de *Expressões Regulares* no BrOffice. Atualmente, existe um bom manual em http://wiki.services.openoffice.org/wiki/Documentation/How_Tos/Regular_Expressions_in_Writer,

Com as recentes mudanças do OpenOffice.org para o LibreOffice, em breve será lançada a versão RC - Release Candidate do LibreOffice, as expressões regulares serão adicionadas a engine como um novo modelo , conforme o link: Wiki TDF

Dito isso, vamos ao que interessa.

Onde usar *Expressões Regulares* no BrOffice

Você pode usar *Expressões Regulares* nas seguintes situações:

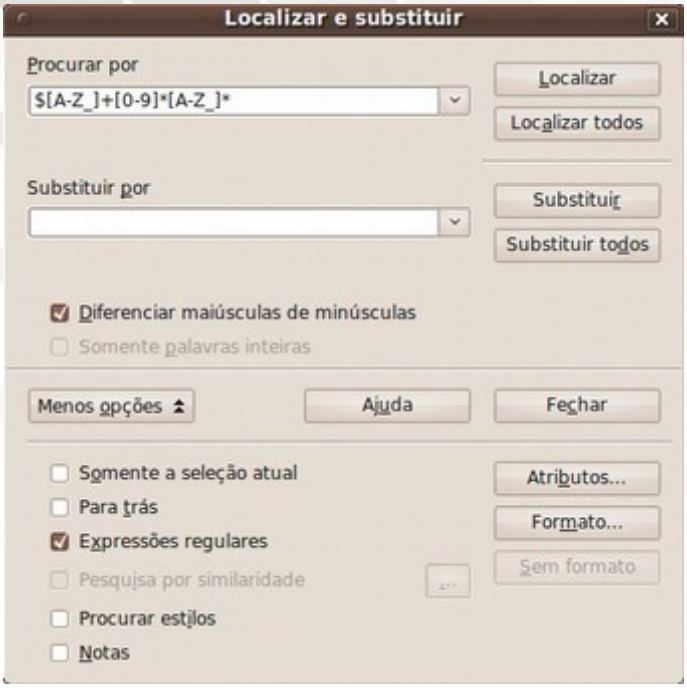
No *Writer*:

- Editar → Localizar e Substituir;
- Editar → Alterações → Aceitar ou Rejeitar → Tab Filtro

No *Calc*:

- Editar → Localizar e Substituir;
- Dados → Filtro → Filtro Padrão & Filtro Avançado;
- Algumas funções como SOMASE, PROCURAR e outras

As caixas de diálogos que aparecem quando você usa os comandos acima, geralmente têm uma opção para usar *Expressões Regulares* (que normalmente está desligada). Por exemplo:



Diferenças lógicas de uso

Existem diferenças de sintaxe com as *Expressões Regulares* que aprendemos, e também existem diferenças na lógica de aplicação. Essas são mais chatas para os veteranos de *Expressões Regulares* sob o *Shell*, pois é necessário uma mudança na forma de raciocinar. Vejam estes casos:

Havia notado um erro no meu livro, mas esqueci de anotar o número da página. Quando fui fazer a alteração, a única coisa que me lembrava era que o erro estava em uma variável do *Shell*. O que fiz? Montei uma *Expressão Regular* começando por cifrão **($)** e casando letras maiúsculas, algarismos e sublinhados **(_),** e estes algarismos não podiam suceder o cifrão **($).** Era assim:

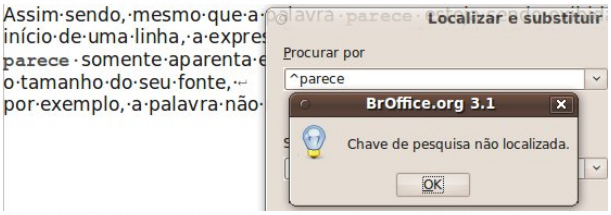
$[A-Z_]+[0-9A-Z_]*

Mas, apesar de certa, a *Expressão Regular* não funcionava pois casava com as variáveis em minusculas também. Demorei muito para descobrir que o *check button* "Diferenciar maiúsculas de minúsculas" é que decide quanto à caixa das letras (veja a figura anterior).

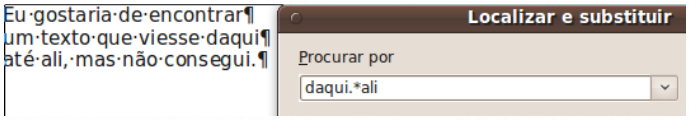
Um outro problema de diferença de lógica, é que no BrOffice as *Expressões Regulares* dividem o texto a ser pesquisado em porções e examinam cada porção separadamente

No *Writer,* o texto é dividido em parágrafos e o que define um parágrafo é a porção de texto entre um **enter** ou um *hard enter* () (caractere não imprimível obtido com **<SHIFT>+<ENTER>** também chamado de *new line*) e outro desses.

Assim sendo, mesmo que a palavra **parece** esteja sendo exibida no início de uma linha, a expressão **^parece** não irá localizá-la, pois **parece** somente aparenta estar no início da linha, mas se alterarmos o tamanho da fonte, por exemplo, a palavra não necessariamente estará mais lá situada.



Por outro lado, a *Expressão Regular* **daqui.*ali,** não irá casar o texto formado pelo **daqui** em um parágrafo até o **ali** em outro. Normalmente os parágrafos são tratados individualmente.



Além disso, o W*riter* considera cada célula de tabela e quadro separadamente. Os quadros são examinados após todos os textos e células de tabela terem sido examinados

Na caixa de diálogo Localizar e Substituir, as *Expressões Regulares* devem ser usadas somente no *box* "Procurar por". Elas não podem ser usadas no "Substituir por:" porque *Expressões Regulares* casam com textos e portanto o que deve ser substituído é o texto e não a *Expressão Regular*.

Diferenças de sintaxe

A partir de agora, veremos as diferenças de sintaxe entre o que aprendemos nas *Expressões Regulares* do Bash e as que veremos do BrOffice.

- Âncoras

Como já vimos existe uma diferença na lógica de uso das Âncoras pois no Bash os *meta caracteres* de início **(^)** e de fim **($)**, procuram respectivamente os textos que os sucedem ou precedem no início ou no fim de uma linha. No BrOffice, esta procura é feita no início ou no fim de um parágrafo.

Podemos também citar um caso que já foi reportado ao BrOffice.org para correção. O **alternativo** ou **ou (|)** pode atrapalhar o início **(^).** Assim, pensando no mengão, se procurarmos **^vermelho|preto,** o BrOffice.org devolverá as ocorrências de **vermelho** iniciando parágrafos ou **preto** em qualquer lugar. Contudo se fizermos: **vermelho|^preto** serão devolvidos o **vermelho** e o **^preto,** ambos em qualquer lugar, desta forma não reconhecendo o *metacaractere* **^** como a Âncora que marca o início de um parágrafo.

Podemos também citar um caso que já foi reportado ao BrOffice.org para correção. O **alternativo** ou **ou (|)** pode atrapalhar o início **(^).** Assim, pensando no mengão, se procurarmos **^vermelho|preto,** o BrOffice.org devolverá as ocorrências de **vermelho** iniciando parágrafos ou **preto** em qualquer lugar. Contudo se fizermos: **vermelho|^preto** serão devolvidos o **vermelho** e o **^preto,** ambos em qualquer lugar, desta forma não reconhecendo o *metacaractere* ^ como a Âncora que marca o início de um parágrafo.

Ué, mas não íamos falar sobre as diferenças de sintaxe? Sim, mas é bom frisar bastante este conceito de parágrafo porque isso algumas vezes me dá uma derrubada.

Dentre as Âncoras, as únicas diferenças de sintaxe que existem no uso de *Expressões Regulares* no Bash e no BrOffice estão nos *meta caracteres* que definem as bordas. As contra barras **(\)** que são *meta caracteres* (*escape*), assumem, na pesquisa uma função diferente quando seguidas por um menor **(\<)** ou um maior **(\>).** Vejamos:

Para procurar palavras que começam por texto:

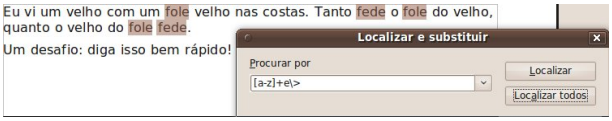
- No Bash fazemos: **\btexto;**
- No BrOffice fazemos: **\<texto.**

Para procurar palavras que terminam por texto:

- No Bash fazemos: **texto\b;**
- No BrOffice fazemos: **texto\>.**

Exemplos:

A *Expressão Regular* **[a-z]+e\>** neste exemplo, foi usada para procuramos por uma ou mais letras **([a-z]+),** terminando com a letra **(e\>).**



Representantes

Quase todos os *meta caracteres* Representante têm uso idêntico ao seus congêneres sob o Bash, porém nem tudo neste mundo é perfeito, porque mesmo que poucas, ainda existem algumas diferenças de uso.

Sob o Bash, todos os caracteres dentro de uma lista **([])** são tratados como literais, exceto o circunflexo que é tratado como um negador da lista (e assim mesmo quando ele é o seu primeiro elemento). Sob o BrOffice usamos a contra barra **(\)** em uma lista para "escapar" alguns alguns meta caracteres *e* também para formar códigos hexadecimais.

Somente os caracteres, abre colchetes **([),** hífen **(-)** e contra barra **(\)** devem ser "escapados" no interior de uma lista, já que nesse ambiente, seus significados são especiais.

Por exemplo a *Expressão Regular* **[[\]a-z]** casa um abre colchetes **([),** ou um fecha colchetas **(]),** ou uma letra minúscula. **[\\]** casa com uma contra barra literal, **[\t]** casa com a letra **'t'.** Para casarmos um **<TAB>,** devemos fazer: **[\x0009]** que é a representação ASCII em hexadecimal do **<TAB>.**

Todos os outros caracteres são tratados pelos seus valores normais, não precisando mais nenhum artifício.

Usando Classes de Caracteres POSIX, surge um caso que também já foi reportado, isso porque uma classe como **[[:alpha:]],** deveria casar com qualquer algarismo decimal, mas isso não se verifica.

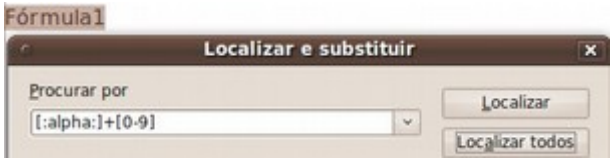
Usando Classes de Caracteres POSIX, surge um caso que também já foi reportado, isso porque uma classe como **[[:alpha:]],** deveria casar com qualquer algarismo decimal, mas isso não se verifica.

Pelo site do BrOffice, este fato reportado funcionaria se fizéssemos **[:digit:],** o que de cara já te traz algumas restrições, como exemplo suponha que eu queira procurar em um arquivo todas as letras minúsculas e todos os algarismos decimais. Nesse caso, sob o BrOffice, não poderia usar Classes de Caracteres POSIX. No Bash faríamos **[[:lower:][:digit:]]** porém isso no BrOffice é impossível, pois em virtude dos colchetes **([])** mais externos, ele encararia isso como uma lista formada todos os caracteres internos a esses colchetes **([])** e não como duas classes.

Da mesma forma, também não funcionaria **[a-z[:digit:]],** nem **[[:lower:]0-9].** A saída seria usarmos uma lista de caracteres, mas assim mesmo, veja só isso:

Viu! Neste caso as letras acentuadas não casaram com o nosso padrão.

Por outro lado, pela facilidade que tenho no uso de *Expressões Regulares*, venho há muito tempo testando-as no BrOffice, e não sei se por erro meu, mas até à versão que estou escrevendo este documento (3.1.1), não consegui casar **[:digit:]** nenhuma vez. Veja:



Duas observações:

Usando a classe [:alpha:], conseguimos casar as letras acentuadas;

A classe [:digit:] não funfou!

Resumindo: essa facilidade ainda está em fase de consolidação e por isso sempre que posso usar as listas convencionais (como no caso de algarismo), não penso duas vezes, uso-as.

Quantificadores

Aqui não há o que temer nem acrescentar. Os Quantificadores do BrOffice se comportam da mesma forma que os do Bash. Poupe seu fôlego para a seção seguinte.

Outros

Aqui que a porca torce o rabo! Nesta classe de *meta caracteres* existem diferenças substancias no uso de *Expressões Regulares*, a começar pelo que já vimos, de uma Âncora sucedendo um **ou (|).** Recapitulando:

Se procurarmos **^vermelho|preto,** o BrOffice devolverá as ocorrências de **vermelho** iniciando parágrafos ou **preto** em qualquer lugar. Contudo se fizermos: **vermelho|^preto** serão devolvidos o **vermelho** e o **^preto,** ambos em qualquer lugar, desta forma não reconhecendo o *meta caractere* ^ como a Âncora que marca o início de um parágrafo.

Vamos ver como funcionam os grupos e os retrovisores: Como já sabemos os grupos são criados com o uso de parênteses. Os **textos** casados (veja bem, são os textos e não as *Expressões Regulares*) ficam guardados e podem ser recuperados dentro da mesma *Expressão Regular*. No exemplo a seguir, procuramos por duas palavras iguais, separadas por um hífen (-). Veja:

A *Expressão Regular* era a seguinte:

\<([a-z]+)-\1\>

Onde:

\< e **\>** Formam a borda esquerda e direita do texto, respectivamente;

([a-z]+) Define uma ou mais letras seguidas (palavra) dentro de um grupo;

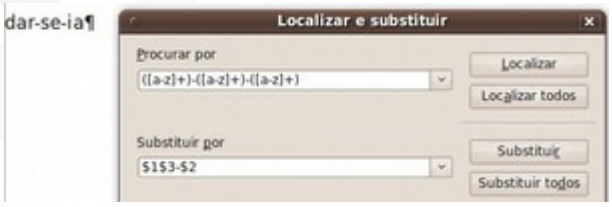
-\1 O hífen seguido do texto casado pelo grupo.

Até agora não mudou nada do que foi dito sobre *Expressões Regulares* no Bash e tudo funcionou às mil maravilhas. Mas se você prestar a atenção, o que fizemos foi clicar em um dos botões de **localizar**. A diferença começa quando desejamos usar o texto casado para **substituir** alguma coisa.

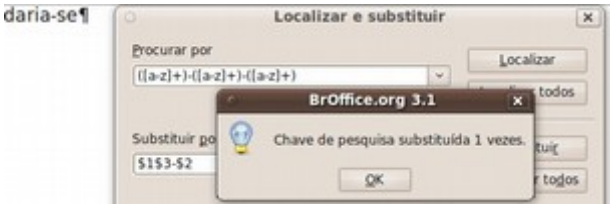
Vamos aproveitar este exemplo para trocar **bate-bate** por **bater:**

Repare que na caixa "Substituir por" eu usei **$1r,** ou seja, o texto casado não é mais o **\1,** agora é o **$1.**

Eu odeio mesóclises e tinha o seguinte texto:



Este **dar-se-ia** estava me torturando, então troquei-o usando uma *Expressão Regular* em que eu montava 3 grupos separados por hífen **(-).** Então, o primeiro **($1)** recebeu **dar,** o segundo **($2)** recebeu **se** e o terceiro **($3)** recebeu **ia.** Colocando-os na ordem que eu queria **($1$3-$2),** veio:



Esta sintaxe pode parecer estranha, mas não é, ela é semelhante a usada na linguagem Perl.

Duas observações sobre o uso de cifrão **($)** para substituir texto:

Se no seu texto tem um valor **123,45** e você desejar substitui-lo por **R$123,45,** você deverá colocar na caixa "Substituir por" **R\$123,45,** veja:

O primeiro grupo casou tudo até a vírgula **(,)** e o segundo ficou com o resto do número. Na substituição, o primeiro cifrão foi precedido por uma contra barra **(\$)** para especificar que era um literal

O $0 na caixa "Substituir por" substitui por todo o texto casado.

A contra barra **(\)** nas *Expressões Regulares* do BrOffice.org também tem suas peculiaridades:

\t Quando fora de uma lista equivale a um **<TAB>.** Como já foi dito, dentro de uma lista, o **<TAB>** deve ser usado com seu código hexadecimal **[\x0009].**

Então para trocar todos os <TAB> por espaços em branco, ponha na caixa "Procurar por" um \t e na caixa "Substituir por" um espaço em branco;

\n Casa com um *hard enter* ou *new line* () (formado por um **<SHIFT>+<ENTER>).**

Suponhamos que você tenha vermelho seguido de um *hard enter* e na linha seguinte preto, se na caixa "Procurar por" colocar vermelho\npreto e na caixa "Substituir por" colocar vermelho-e-preto, após a substituição, o *hard enter* terá morrido, sobrando vermelho-e-preto. O mesmo poderia ter sido feito, colocando nas caixas **\n** e **-e-,** respectivamente;

$ Casa com uma marca de parágrafo **().**

Diferentemente do *hard enter*, não podemos definir na caixa "Procurar por" a cadeia no início da linha seguinte um uma *Expressão Regular* que case com ela. Assim sendo, não poderíamos agir como no exemplo anterior, preenchendo as caixas com vermelho\npreto e com **vermelho-e-preto** respectivamente, porém funcionaria se as caixas fossem preenchidas com **$** e com **-e-.**

\x Quando temos um **\xNNNN,** onde **NNNN é um** algarismo hexadecimais **[0-9A-Fa-f]** na caixa "Procurar por", o BrOffice localizará (e eventualmente trocará) o caractere definido pelo código hexadecimal formado por **NNNN.** Se este código estiver na caixa "Substituir por", será tratado como um literal.

Isso me lembra **um** macete de edição que uso. Ao longo de um texto grande, escrevemos diversos ordinais como **2ª.** Isso pode ser feito em **Inserir → Caractere especial,** mas fazer isso um monte de vezes num texto, enche o saco. Como faço? Para fazer **2ª,** como disse, escrevo **2.a.** e assim vou fazendo com todos os ordinais até terminar de escrever o documento.

Ao final, coloco **na** caixa "Procurar por" a *Expressão Regular* **([0-9])\.a\.,** e na caixa "Substituir por" coloco um **$1ª,**ou seja, o texto que casou com o grupo (um algarismo) seguido do ordinal. Em seguida clico em "Substituir todos". Repito mais uma vez esta operação para substituir os ordinais masculinos como em **12º**

Vou terminar este texto com uma dica que demorei muito para descobrir. Ela é muito útil quando se copia um texto de um navegador e se cola no BrOffice. Normalmente o nove texto fica com um monte de *hard enter*, não é?

Como vocês viram ainda existem algumas coisas a serem acertadas e outras a serem feitas, mas o concorrente direto do BrOffice, o MS Office, nem sonha ter *Expressões Regulares*. Creio que essas pequenas alterações e incrementos sejam sanados com a nova versão (bastante modificada), descrita em http://userguide.icu-project.org/strings/regexp.