E-fólio A | Folha de resolução para E-fólio

UNIDADE CURRICULAR: Liguagens e Computação

CÓDIGO: 21078

DOCENTE: Prof. Jorge Morais

A preencher pelo estudante

NOME: Marc Paulo Martins

N.º DE ESTUDANTE: 1601202

CURSO: Licenciatura em Informática

DATA DE ENTREGA: 19/11/2017

TRABALHO / RESOLUÇÃO:

Questão 1: Construa e teste expressões regulares em notação UNIX para localizar anos bissextos:

Antes de mais, fomos ler na Wikipédia, o que é um ano bissexto: “Chama-se ano bissexto o ano ao qual é acrescentado um dia extra, ficando ele com 366 dias, um dia a mais do que os anos normais de 365 dias, ocorrendo a cada quatro anos (exceto anos múltiplos de 100 que não são múltiplos de 400). Isto é feito com o objetivo de manter o calendário anual ajustado com a translação da Terra e com os eventos sazonais relacionados às estações do ano. O anterior ano bissexto foi 2016 e o próximo será 2020.”

Vamos então por etapas, com expressões regulares intermediárias (Regex) para chegar ao regex final. Queremos que o regex final seja **válido para qualquer ano (a partir do ano 0 e sem limite superior)**:

- A enorme maioria dos anos acaba por 04, 08, 12, 16… Uma vez que este evento ocorre cada 4 anos (fora das excepções que serão tratadas a seguir). Assim o primeiro regex que surge é:

[0-9]*(04|08|12|16|20|24|28|32|36|40|44|48|52|56|60|64|68|72|76|80|84|88|92|96)

[0-9]*: qualquer algarismo entre 0 e 9, o número de vezes necessária (pode ser 0 vez).

(04|08…|96): o ano deve obrigatoriamente acabar com um destes dois números.

- De seguida, vamos começar a tratar o caso excluída da excepção, isto é anos que sejam múltiplos de 400. Acrescentemos ao regex supra o ‘00’ opcional:

[0-9]*(04|08|12|16|20|24|28|32|36|40|44|48|52|56|60|64|68|72|76|80|84|88|92|96)(00)?

No entanto, este regex seria válido para qualquer ano múltiplo de 100 e não apresenta os números 0, 4, 8 que possam contemplar os anos 0, 4, 8. Vamos então acrescentar estas possibilidades no início do regex:

(0|4|8|[0-9]*(04|08|12|16|20|24|28|32|36|40|44|48|52|56|60|64|68|72|76|80|84|88|92|96)(00)?)

- Após uns testes no www.regexpal.com, falta-nos o caso dos anos 10000, 100000, 1000000.... Assim, ao adicionar |10000|, obtemos o regex que contempla todos os casos:

(0|4|8|10000|[0-9]*(04|08|12|16|20|24|28|32|36|40|44|48|52|56|60|64|68|72|76|80|84|88|92|96)(00)?)

Questão 2: Construa e teste expressões regulares em notação UNIX para localizar número inteiros em notação binária que sejam múltiplos de 4 mas não sejam múltiplos de 8:

Vamos por etapas:

Um múltiplo de 4 em binário é qualquer número binário que termina em 00.

Assim o regex neste caso seria:

[10]*00

Um múltiplo de 8 em binário é qualquer número binário que termina em 000. Assim, temos que acrescentar no regex anterior, uma excepção para este caso. Uma vez que o múltiplo inferior será 100, temos necessariamente número com 3 bits. Assim, o regex seguinte permitirá excluir sempre os múltiplos de 8, isto é, que terminam por 000:

[10]*100

Questão 3: Construa e teste expressões regulares em notação UNIX para localizar endereços de e-mail:

Uma primeira solução trivial seria:

[A-Za-z0-9._-]+@[A-Za-z0-9._-]{2,}\.[A-Za-z]{2,}

No entanto, este regex bastante comum não cobre a esmagadora maioria dos casos. Fomos então, novamente, pesquisar os elementos normativos que regem a sintaxe de um e-mail. No wikipédia, encontrámos a informação seguinte (em inglês):

“The format of email addresses is local-part@domain where the local part may be up to 64 characters long and the domain may have a maximum of 255 characters. The formal definitions are in RFC 5322 (sections 3.2.3 and 3.4.1) and RFC 5321—with a more readable form given in the informational RFC 3696[3] and the associated errata. Note that unlike the syntax of RFC 1034 and RFC 1035, there is no trailing period in the domain name.

Local-part

The local-part of the email address may use any of these ASCII characters:

- uppercase and lowercase Latin letters A to Z and a to z;

digits 0 to 9;

- special characters !#$%&'*+-/=?^_`{|}~;
- dot ., provided that it is not the first or last character unless quoted, and provided also that it does not appear consecutively unless quoted (e.g. John..Doe@example.com is not allowed but "John..Doe"@example.com is allowed)
- Note that characters after a plus sign + are generally ignored, so fred+bah@domain and fred+foo@domain will end up in the same inbox as fred@domain This can be useful for tagging emails for sorting, see below.

- space and "(),:;<>@[\] characters are allowed with restrictions (they are only allowed inside a quoted string, as described in the paragraph below, and in addition, a backslash or double-quote must be preceded by a backslash);
- comments are allowed with parentheses at either end of the local-part; e.g. john.smith(comment)@example.com and (comment)john.smith@example.com are both equivalent to john.smith@example.com.
- In addition to the above ASCII characters, international characters above U+007F, encoded as UTF-8, are permitted by RFC 6531, though mail systems may restrict which characters to use when assigning local-parts.

Domain

The domain name part of an email address has to conform to strict guidelines: it must match the requirements for a hostname, a list of dot-separated DNS labels, each label being limited to a length of 63 characters and consisting of:[9]

uppercase and lowercase Latin letters A to Z and a to z;

digits 0 to 9, provided that top-level domain names are not all-numeric;

hyphen -, provided that it is not the first or last character.”

Neste exercício, não vamos contemplar os casos de um email com comentários, com espaços, ( ) , : ; [\ ] < > @ (excepto o @ de separação), e o caso do + seguido de um ou mais caracteres uma vez que não são reconhecidos ou então autorizados com restrições por alguns servidores.

Vamos então por passos:

A parte local é constituída dos caracteres ASCII acima autorizados. Podemos definir o regex para esta parte local:

[a-z0-9_\-!#$%&'*+-/=?^_`{|}~]+

Note-se \- e * para que possa tido em conta o carácter especial -. No entanto, este regex não contempla os pontos e o caso deles não puderem repetir-se consecutivamente. Assim colocamos esta excepção no início do regex anterior:

(?!.*\.{2})[a-z0-9_\-!#$%&'*+-/=?^_`{|}~]+

No entanto o ponto não pode encontrar-se no início ou no fim da parte local, assim:

[a-z0-9_!#$%&'*+\-/=?^_`{|}~]((?!.*\.{2})[a-z0-9_!#$%&'*+\-/=?^_`\.{|}~][a-z0-9_!#$%&'*+\-/=?^_`{|}~]+)*

Assim, este regex para a parte local contempla as excepções previstas para o ponto. E assim também para a parte local.

Relativamente ao domínio, e conforme indicado supra, o regex é:

([A-Za-z0-9._]([A-Za-z0-9._\-][A-Za-z0-9._])*)\.[a-z]{2,}

Temos obrigatoriamente um carácter inicialmente (que não inclui o hífen) seguido do domínio que deve ter pelo menos 2 caracteres.

Finalmente, o regex do nosso e-mail será:

[a-z0-9_!#$%&'*+\-/=?^_`{|}~]((?!.*\.{2})[a-z0-9_!#$%&'*+\-/=?^_`\.{|}~][a-z0-9_!#$%&'*+\-/=?^_`{|}~]+)*@([A-Za-z0-9._]([A-Za-z0-9._\-][A-Za-z0-9._])*)\.[a-z]{2,}

Sabendo que este regex não tem em conta os constrangimentos de comprimentos e dos caracteres não reconhecidos por todos os servidores (ver acima).

FIM