Regex Marathon :

(Correction : les réponses ou commandes egrep à saisir sont en dessous de chacune des 15 questions. Bien sûr, d'autres motifs peuvent parvenir à des résultats similaires)

1/ Rechercher l'ensemble des lignes contenant le mot : "define"

--> commande: egrep define rfc5234.txt ; // plusieurs lignes de texte s'affichent

2/ Combien de fois apparait ce mot (vous pouvez utiliser le paramètre -c de la commande ou compter à la main :)

--> commande: egrep -c define rfc5234.txt ; 10

Note : Nous allons au cours des 6 prochains items découvrir les éléments fondamentaux du langage des expressions régulières

3/ Les **classes de caractères**, définies entre [], qui permettent de chercher une correspondance pour un ensemble de caractères, par exemple de [a-zA-Z]. Plusieurs classes de caractères sont prédéfinies ou bénéficient de raccourci ([:alpha:], [:alnum:], [:space:], \w). Par ailleurs le caractère "." correspond à n'importe quel caractère (espaces et ponctuations comprises). On peut définir une classe de caractère par son complémentaire grâce au caractère ^, c'est à dire par la négative : [^a-z], les caractères qui ne sont pas entre 'a' et 'z'.

Recherchez l'ensemble des lignes contenant une date au format YYYY (il y en a 20 dans le document).

--> [0-9] [0-9] [0-9] [0-9] mais pour ce document c'est imparfait, [12] [09] [0-9] [0-9] par exemple

4/ Les **quantificateurs** qui permettent définir le nombre de répétitions de tout ou partie d'un motif : * (0 ou n fois), + (au moins 1 fois), ? (0 ou une fois), {n,m} (un nombre précis d'occurrences entre n et m).

Recherchez les lignes contenant des entiers ayant plus de deux chiffres (il y en a 30).

--> [0-9] [0-9]+, par exemple

Recherchez l'ensemble des lignes contenant au moins un mot de 15 lettres (il y en a 11).

--> [[:alpha :]]{15}, par exemple

5/ Des **marqueurs de début et fin** permettent d'indiquer si le motif doit se trouver en début, '^' ou en fin de ligne '$'. '\<' et '\>' pour marquer respectivement le début ou la fin d'un mot (en tant que succession de caractères séparé par un caractère d'espacement.

Recherchez les lignes se terminant par un chiffre (il y en a 62).

--> [0-9]$, par exemple

6/ Les **alternatives** avec le caractère | permet d'exprimer une succession de motifs alternatifs (foo| bar, correspond à soit la chaîne "bar", soit la chaîne "foo", soit une combinaison des deux.

Recherchez les lignes qui contiennent "foo" ou "bar" dans le document (il y en a 12)

--> foo\>|bar\> serait assez précise pour ne pas capturer de bruit

7/ Les **regroupements** qui avec les () permettent de définir des sous-motifs qui permettent de segmenté le motif (exemple : aaa(b|c), c'est à dire 'aaab' ou 'aaac'). Il est possible de faire référence aux sous-motifs dans leur ordre d'apparition dans l'expression régulière (\1, \2, ...). Par exemple, le motif "([a-zA-Z]{3,}) \1" peut être utilisé pour détecter de potentielles erreurs de répétitions (le même mot répété deux fois à la suite dans la même ligne – il n'y en a pas dans la rfc5234).

Recherchez les lignes contenant deux fois le mot "foo" ou le mot "blat" en utilisant le mécanisme de référence à un sous motif (il y en a 2).

--> (foo|blat).*\1, par exemple

8/ Echappement des caractères : Si l'on souhaite chercher pour un des caractères utilisés par le langage des expressions régulières, il faut alors faire un échappement avec '\'.

Recherchez toutes les lignes contenant des expressions entre parenthèses.

--> \(.+\)

Le principal intérêt des expressions régulières est de permettre de rechercher des variations de motifs de caractères (et par extension de mots). Les expressions régulières nous offre un langage riche pour exprimer des motifs. Par exemple :

egrep "^[[:digit:]].*$" file.txt retournera l'ensemble des lignes de file.txt qui commencent par un chiffre.

egrep "^[^0-9].*$" file.txt retournera l'ensemble des lignes de file.txt qui ne commencent pas par un chiffre.

Note : Maintenant que vous avez les bases, vous pouvez poursuivre avec les questions suivantes.

9/ Recherchez toutes les lignes commençant par Internet ou RFC (attention, si vous observez bien le document, la plupart des lignes commencent par des caractères d'espacement). Indication : il y en a 21 exactement.

--> egrep "^[[:space:]]*(Internet|RFC)" rfc5234.txt

10/ En utilisant le quantificateur +, recherchez tous les mots qui commencent par 'def' (mais ne sont pas exactement la chaîne 'def'). Utilisez le paramètre –o pour ne voir que les caractères qui correspondent à ce que votre expression régulière capture.

--> egrep "\<def\w+\>" rfc5234.txt, (remarque, du point de vue des lignes retournées egrep "def" donne le même résultat mais la question 8 les dirige de toute façon vers cette solution.)

11/ Le mot 'def' exactement est-il présent dans le document ? Définissez une expression régulière permettant de le vérifier

--> Non, egrep "\<def\>" rfc5234.txt (rechercher def en l'entourant par des caractère espaces est une solution alternative)

12/ Utilisez le paramètre -o pour voir uniquement les séquences de caractères correspondant à l'expression régulière définie en 6.

Essayer de n'extraire qu'un seul mot à la fois (et non la ligne entière). Faîtes afficher l'ensemble des variations des mots commençant par 'def' du document.

--> egrep -o "\<def\w*\>" rfc5234.txt

13/ Les expressions régulière sont-elles sensibles à la casse des caractères ?

--> Par défaut oui, il faut rajouter le paramètre -i pour que ce ne soit pas le cas.

14/ Dans rfc5234.txt, l'ensemble des références bibliographiques sont marquées entre '[' et ']'. Définissez un motif permettant d'extraire les références bibliographiques du document quand elles sont citées dans le texte et à la fin du document (et seulement celles-ci).

--> egrep "\[[[:alpha:]]+(-[[:alpha:]]+|[[:digit:]]+)\]" rfc5234.txt

15/ Définissez une expression régulière qui permettent d'extraire toutes les lignes se terminant par un mot écrit en majuscule.

--> egrep "[A-Z]+$" rfc5234.txt

16/ Définissez et mettez en application une expression régulière permettant d'extraire les lignes des règles de grammaire au format ABNF du document (et seulement celle-ci)

--> egrep "^[[:space:]]{6,}.*=.*$" rfc5234.txt

17/ Définissez et mettez en application une expression régulière permettant d'extraire toutes les adresses email du document (avec le paramètre -o)

--> egrep "[[:alnum:]\-\.]+@[[:alnum:]]+\.[[:alpha:]]{2,3}"

// Ils peuvent essayer de la rendre la plus générique possible mais de toute façon il n'y a pas d'expression régulière parfaite à ce niveau.

18/ Supposons que l'on veuille préparer une version de rfc5234.txt sans pagination (c'est à dire en supprimant les entête et pieds de page), en analysant la forme du document (espacement, indentation…) définissez une expression régulière permettant de le faire avec egrep.

--> egrep "^([[:digit:]]|[[:space:]]+).+$" rfc5234.txt

3. Pour aller plus loin

Afin de mieux comprendre comment les expressions régulières peuvent être utilisées en javascript, allez sur tryregex.com et complétez le tutoriel interactif suivant :

http://tryregex.com/