Feuille de travaux pratiques n° 1
Utilisation des expressions régulières

Exercice 1.1 (Compter les mots)

Télécharger un texte (par exemple le fichier Germinal depuis Madoc) et ranger ce texte (`nomfic.txt`) dans le répertoire de travail. Compter les mots de ce texte de différentes manières :

1. En utilisant la commande unix `wc` dans un terminal après avoir rendu actif le répertoire de travail avec la commande `cd`. Pour savoir comment l'utiliser utiliser la commande : `wc -help`.
2. En chargeant le texte dans l'*éditeur de texte gedit et en utilisant* Outils → Statistiques du document.
3. En chargeant le texte dans le *traitement de textes Free Office et en utilisant* Outils → Statistiques.
4. En écrivant un programme dans un langage de programmation qui lit un flux de caractères (les caractères d'un fichier), qui sépare les caractères en mots et qui compte ces mots.

Comparer les résultats.

Exercice 1.2 (Expressions régulières)

Extrait du `man grep`
Une expression rationnelle (regular expression) est un motif qui permet de décrire un ensemble de chaînes. Les expressions rationnelles sont construites comme des opérations arithmétiques ; elles utilisent différents opérateurs pour combiner des expressions plus petites.
Les briques élémentaires sont les expressions rationnelles correspondant à un seul caractère. La plupart des caractères, y compris les lettres et les chiffres, sont des expressions rationnelles qui concordent avec eux-mêmes. Tout méta-caractère ayant une signification spéciale doit être protégé en le faisant précéder d'une contre-oblique (backslash).
Une liste de caractères, encadrée par `[` et `]` peut être mise en correspondance avec n'importe quel caractère appartenant à la liste. Si le premier caractère de la liste est l'accent circonflexe `^` alors la mise en correspondance se fait avec n'importe quel caractère absent de la liste. Par exemple, l'expression rationnelle `[0123456789]` concorde avec n'importe quel chiffre.
Entre ces crochets, un intervalle de caractères peut être indiqué en donnant le premier et le dernier caractère, séparés par un tiret. Il correspond à n'importe quel caractère compris entre le premier et le dernier caractère (ceux-ci inclus), l'ordre des caractères dépendant des paramètres régionaux (locale, en anglais) en cours. Ainsi avec la valeur par défaut (appelée "C"), `[a-d]` est équivalent à `[abcd]`. Avec beaucoup de paramètres régionaux, les caractères sont triés en suivant l'ordre des dictionnaires, et `[a-d]` n'est alors pas équivalent à `[abcd]`, mais à `[aBbCcDd]`, par exemple. Pour que ces listes aient le comportement usuel de C, vous pouvez positionner la variable d'environnement `LC_ALL` à la valeur C.
Enfin, il existe des classes de caractères prédéfinies. Leurs noms sont assez explicites : `[:alnum:]`, `[:alpha:]`, `[:cntrl:]`, `[:digit:]` (chiffres), `[:graph:]`, `[:lower:]` (minuscules), `[:print:]` (c. affichables), `[:punct:]` (ponctuation), `[:space:]` (espace), `[:upper:]` (majuscules), et `[:xdigit:]` (chiffres hexadécimaux). Par exemple, `[[:alnum:]]` correspond à `[0-9A-Za-z]`, à la différence près que le dernier dépend des paramètres régionaux C et du codage de caractères ASCII, alors que le premier est plus portable. Remarquez que les crochets dans les noms de classes font partie intégrante du nom symbolique, et qu'ils doivent donc être inclus en plus des crochets encadrant la liste. La plupart des méta-caractères perdent leur signification spéciale au sein des listes. Pour inclure un caractère `]`, mettez-le en premier dans la liste. De même, pour inclure un caractère `^`, placez-le n'importe où sauf au début de la liste. Enfin, pour inclure un `-`, placez-le en dernier.

Le point `.` correspond à n'importe quel caractère. Le symbole `\w` est un synonyme de `[[:alnum:]]` et `\W` un synonyme de `[^[:alnum]]`.
L'accent circonflexe `^` et le symbole dollar `$` sont des méta-caractères correspondant respectivement à une chaîne vide au début et en fin de ligne. Les symboles `\<` et `\>` correspondent respectivement à une chaîne vide en début et en fin de mot. Le symbole `\b` correspond à une chaîne vide à l'extrémité d'un mot, et `\B` correspond à une chaîne vide ne se trouvant pas à une extrémité de mot.
Une expression rationnelle correspondant à un caractère unique peut être suivie par l'un des opérateurs de répétition suivants :
`?` L'élément précédent est facultatif et peut être rencontré au plus une fois.
`*` L'élément précédent peut être rencontré zéro ou plusieurs fois.
`+` L'élément précédent peut être rencontré une ou plusieurs fois.
`{n}` L'élément précédent doit être cherché exactement n fois.
`{n,}` L'élément précédent doit être cherché n fois ou plus.
`{n,m}` L'élément précédent doit être cherché au moins n fois, mais au plus m fois.
Deux expressions rationnelles peuvent être juxtaposées ; l'expression résultante correspondra à toute chaîne formée par la juxtaposition de deux sous-chaînes correspondant respectivement aux deux expressions.
Deux expressions rationnelles peuvent être reliées par l'opérateur infixe `|` ; l'expression résultante correspondra à toute chaîne concordant avec l'une ou l'autre des deux expressions.
Les répétitions ont priorité sur les juxtapositions, qui à leur tour ont priorité sur les alternatives. Une sous-expression peut être entourée par des parenthèses pour modifier ces règles de priorité.
Rechercher dans le texte "germinal.html"

1. tous les mots qui commencent par "p" ou "P" et terminent par "e",
2. tous les nombres,
3. toutes les formes du verbe "regarder",
4. tous les mots qui commencent par une majuscule,
5. tous les mots qui contiennent un double "t" comme "attendre",
6. tous les mots qui ont entre 10 et 20 lettres,
7. toutes les occurrences du mot "est" et leurs contextes gauche et droit de 21 caractères.

Peut-on (et dans ce cas avec quelle expression rationnelle) rechercher

1. les entiers impairs,
2. les formes bien parenthésées comme `Pierre (un homme (très) grand)`,
3. les suites de nombres croissants ?

1) Avec la commande unix `grep`
On travaille dans un terminal. Il faut rendre actif le répertoire dans lequel se trouvent les textes à analyser, puis utiliser la commmande ci-dessus avec l'expression régulière appropriée.

Syntaxe : `grep -Eon expression-régulière nom-du-fichier`
Signification des options : `E` recherche une expression régulière, `o` affiche seulement l'occurrence correspondant à l'expression régulière pas toute la ligne, `n` affiche les numéros de lignes où sont trouvés les motifs. Pour connaître toutes les options de la commande grep : `grep -help`

Par exemple : `grep -Eon ' a[a-z]*t ' conte_fr.txt`
affiche tous les mots (entre deux espaces) qui commencent par "a" et se terminent par "t". Chaque mot est précédé du numéro de la ligne à laquelle il appartient dans le texte. Pour trouver les mots qui sont limités par d'autres caractères que l'espace (apostrophe, ponctuation ...), on peut remplacer l'espace par le caractère `\b`.

2) Avec le traitement de textes de Free Office
— Charger le texte à analyser dans le traitement de texte.
— *Édition* → *Rechercher & Remplacer. Ouvrir* Autres Options et cocher "Expressions régulières".
— Écrire l'expression régulière dans le champ *Rechercher*
— *Activer le bouton* Rechercher ou *Rechercher tout*