5 - Expressions régulières

iode conseil
ingénierie de la formation à distance

- Principe
- Motifs de recherche
- Opérateur de correspondance
- Opérateur de substitution
- Opérateur de translittération
- Fonctions `split` et `join`

Principe

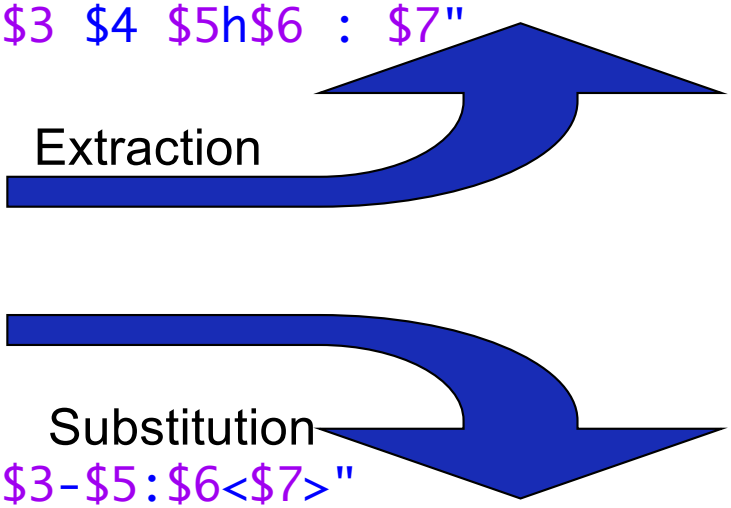
Motif de recherche :

```
/^-\s*(de|entre)?\s*(\d?\d)h(\d\d)
 \s*(à|et)\s*
 (\d?\d)h(\d\d)\s*:\s(.*)$/i
```

Programme :

- De 9h00 à 9h30 : petit déjeuner (pensez à commander les croissants la veille)
- 9h30 à 12h00 : réunion de travail
- entre 12h15 et 14h00 : déjeuner (au bistro du coin)
- de 14h15 à 17h00 : suite réunion

9h00 à 9h30 : petit déjeuner
9h30 à 12h00 : réunion de travail
12h15 et 14h00 : déjeuner
14h15 à 17h00 : suite réunion



09:00-09:30<petit déjeuner>
09:30-12:00<réunion de travail>
12:15-14:00<déjeuner>
14:15-17:00<suite réunion>

Motifs de recherche 1/12

iode conseil
ingénierie de la formation à distance

- Expression régulière : motif à comparer à une chaîne donnée, comportant une suite d'assertions, d'atomes (assertions de largeur non nulle) quantifiables, chacune pouvant être :
  - des caractères normaux : *c* ou `\m`
  - des métacaractères *m* : `\ | ( ) [ { ^ $ * + ? .`
    - simples : ex. : un caractère quelconque, le début de la chaîne
    - structurels : ex. : *n* fois le caractère qui précède
  - des métasymboles : `\c` : ex. :
    - une tabulation (largeur 1),
    - une frontière de mot (largeur 0)

Motifs de recherche 2/12

iode conseil
ingénierie de la formation à distance

- Atomes correspondant à un caractère 1/5 :
  - Caractères simples et métasymboles jokers (., \X, \C):

```
"janvier" =~ m/anv/,
"juin", "juillet" =~ /j.i/,
```

  - caractères spécifiques :

Symboles	Signification	Symboles	Signification
\0	Caractère nul	\n	Saut de ligne
\a	Alarme	\r	Retour chariot
\e	Échappement	\t	Tabulation
\f	Fin de page		

Motifs de recherche 3/12

iode conseil
ingénierie de la formation à distance

- Atomes correspondant à un caractère 2/5 :
  - Autres caractères spécifiques :
    - \c*X* : caractère Control-*X*, *X* ∈ {C, Z, [(Esc), ? (Del)}
    - *NNN* : caractère en code octal (0 initial conseillé)
    - \x*HEX* : caractère en code hexa comportant 1 (si car suivant non hexa) ou 2 chiffres hexa
    - \x{*HEXLONG*} : caractère en code hexa *n* chiffres hexa
    - \N{*NOM*} : caractère Unicode nommé : ex. : \N{epsilon} (module charnames requis)

Motifs de recherche 4/12

iode conseil
ingénierie de la formation à distance

- Atomes correspondant à un caractère 3/5 :
  - classes de caractères :

```
"i"          =~ /[aeiouy]/,
","          =~ /[,;.]/,
"4"          =~ /[01234567 89]/, /[0-9]/,
"5", "-"     =~ /[0-9\-]/,
"B"          =~ /[a-zA-Z0-9_]/,
"8"          !~ /[^0-9]/,
"[", "]"     =~ /[[a-z\]]/,
"^"          =~ /[\^0-9]/ , /[0-9^]/
```

Motifs de recherche 5/12

iode conseil
ingénierie de la formation à distance

- Atomes correspondant à un caractère 4/5 :
  - raccourcis de classes de caractères :

Symbole	Signification	Classe équivalente	Négation
\d	chiffre	[0-9]	\D
\w	caractère de mots	[a-zA-Z0-9_]	\W
\s	caractère espace	[\r\t\n\f]	\S

  - classes de caractères de style POSIX : [:*CLASSE*:], à utiliser dans la construction d'une classe de caractère : ex :

```
/[.,[:alpha:][:digit:]]/
/[[:^digit:]]/
```

Motifs de recherche 6/12

iode conseil
ingénierie de la formation à distance

- Atomes correspondant à un caractère 5/5 :
  - propriétés Unicode : \p{*PROP*} et leur négation \P{*PROP*} : ex. :
    - propriétés Unicode standards et composites :

```
/\p{IsLl}/          # lettre minuscule
/\p{IsDCsuper}/     # position d'exposant
/\p{IsAlpha}/       # alphabétique
/\p{Digit}/         # chiffre décimal
/\pN/               # car. numérique (ex. romain)
…
```

    - propriétés de bloc Unicode :

```
/\p{InBengali}/
/\p{InTibetan}/
…
```

Motifs de recherche 7/12

iode conseil
ingénierie de la formation à distance

- Parenthèses de mémorisation/capture et regroupement :
  - $i capture les (..) successifs :

```
/...(..)..(..)....(..).../
#     $1      $2        $n
```

  - \i spécifie une référence arrière
  - Exemples :

```
"..33-44.."  =~ /(\d)\1-(\d)\2/;
# => $1 vaut "3", $2 vaut "4"
"<U V>" =~ /((\w) (\w))/;
# => $1 vaut "U V",
#    $2 vaut "U", $3 vaut "V"
```

Motifs de recherche 8/12

iode conseil
ingénierie de la formation à distance

- Quantificateurs d'atomes ou de regroupement d'atomes 1/2 :

Maximum	**Minimum**	**Nombre d'occurrence N**
{*MIN*, *MAX*}	{*MIN*, *MAX*}?	*MIN* <= N <= *MAX*
{*MIN*,}	{*MIN*,}?	*MIN* <= N
{*NB*}	{*NB*}?	N = *NB*
*	*?	N >= 0
+	+?	N >= 1
?	??	0 <= N <= 1

- **Minimum** signifie que, pour un même point de départ dans la chaîne (le plus à gauche étant toujours privilégié), la correspondance minimale est la moins gloutonne

Motifs de recherche 9/12

iode conseil
ingénierie de la formation à distance

- Quantificateurs d'atomes 2/2 : exemples :

```
"abd", "abcccd" =~ /abc*d/,
"<abcabc>"      =~ /<(abc)*>/,
"/usr/bin"      =~ m#/?\w+(/\w+)*#,
"0 801 802 803" =~ /\d{3}/,
# $1 prend ci-dessous différentes valeurs
"exigence" =~ /e(.*)e/,    # => "xigenc"
"exigence" =~ /e(.*?)e/,   # => "xig"
"exigence" =~ /.*e(.*?)e/, # => "nc"
```

Motifs de recherche 10/12

iode conseil
ingénierie de la formation à distance

- Motifs/Assertions d'ancrage :
  - permettent de lier certaines parties d'un motif à des positions particulières de la chaîne
  - début de chaîne : assertions ^ et \A :

```
"0561..." =~ /^05/
```

  - fin de chaîne : assertions $, \Z et \z :

```
"latin", "patin\n" =~ /tin$/
```

  - limites de mots (\w+) : assertions \b et \B :

```
"c'est haut" =~ /\best\b/
"reste"      =~ /\Best\B/
"estival"    =~ /\best\B/
"ouest"      =~ /\Best\b/
```

  - Mais, dans [\b], \b représente un car. espace arrière

Motifs de recherche 11/12

iode conseil
ingénierie de la formation à distance

- Alternative 1/2 :
  - | permet de faire correspondre un motif ou un sous-motif à un ensemble de possibilités
  - Pour une même position de départ (variant de gauche à droite), les éléments de l'alternative sont cherchés du premier au dernier
  - Exemples :

```
/bleu|vert|rouge/
/(bleu|vert) clair/
"vente", "verte", "veste" =~ /ven|r|ste/
                             /ve(n|r|s)te/
"chameau" =~ /chameau(x|)/
```

Motifs de recherche 12/12

iode conseil
ingénierie de la formation à distance

- Alternative 2/2 :
  - Exemples (suite) :

```
"Par Paris" =~ /(Paris|Par)/,
# => $1 vaut "Par"
"Par Paris" =~ /.*(Paris|Par)/,
# => $1 vaut "Paris"
"Par Paris" =~ /(Paris|Par)$/,
# => $1 vaut "Paris"
/^a|b|c$/ diffère de
/^(a|b|c)$/ et /^a$|^b$|^c$/
```

Opérateur de correspondance 1/3

iode conseil
ingénierie de la formation à distance

- Selon qu'il y a correspondance ou pas,
  chaine =~ m/*motif*/ ou
  m/*motif*/ (*$_* =~ m/*motif*/) renvoie selon le contexte :
  - scalaire : « vrai » ou « faux »
  - liste : ($1, $2, …, $n)
- m/*motif*/ peut-être simplifié en /*motif*/ ou modifié en m#*motif*#, m[*motif*], m{*motif*}… (cf. q/…/)
- *motif* interpole les variables et séquences d'échappement
- De plus, *chaine* =~ "$`$&$'" où $& est la sous-chaîne en correspondance
- Modificateurs de motif : m/*motif*/**i** s m x **o** g cg :
  - /i : ignorer la casse
  - /o : compiler le motif une seule fois

Opérateur de correspondance 2/3

iode conseil
ingénierie de la formation à distance

- Exemples 1/2 :

```
$a = "Ca fait 128h34M27s !";
$cc = '\d\d';
if ($a =~ /\d+h${cc}m${cc}s/i) {
    print "<$`><$&><$'>\n";
    # "<Ca fait ><128h34M27s>< !>\n"
}

($h, $m, $s) =
    $a =~ /(\d+)h($cc)m($cc)s/i;
```

Opérateur de correspondance 3/3

iode conseil
ingénierie de la formation à distance

- Exemples 2/2 :

```
$a = "C'est H 13";
$h = "h";
if ($a =~ /est \u$h $cc/) {
    # vrai
}

$from = "From:";
while (<>) {
    if (/$from (.*)/o) {
        # traiter $1
    }
}
```

Opérateur de substitution 1/3

iode conseil
ingénierie de la formation à distance

- Selon qu'il y a correspondance ou pas, *lvalue* =~ s/*motif*/*subst*/ ou s/*motif*/*subst*/ ($_ =~ s/*motif*/*subst*/) substitue dans *lvalue* la chaîne correspondant au motif par *subst* et retourne (en contexte scalaire ou liste) le nb. de substitutions réalisées ou « faux »
- *subst* est interpolée et $1, $2, …, $n peuvent y être utilisées
- s/*motif*/*subst*/ peut-être modifié en s#*motif*#*subst*#, s[*motif*]{*subst*}… (cf. q/…/)
- Modificateurs de motif : m/*motif*/**i** s m x o **g** e :
  - /i : ignorer la casse
  - /g : remplacer toutes les occurences

Opérateur de substitution 2/3

iode conseil
ingénierie de la formation à distance

- Exemples 1/2 :

```
$a = "Ca fait 128h34M27s !";
if ($a =~ s/Ca fait/->/) {
    for ($a) {
        s/^\W*//;
        s/\W*$//;
        s/(\d+)/<$1>/g;
        s/(\D)/\u$1/g;
    }
}
# => $a vaut "<128>H<34>M<27>S"
```

Opérateur de substitution 3/3

iode conseil
ingénierie de la formation à distance

- Exemples 2/2 :

```
$m1 = "11h22";
($m2 = $m1) =~ s/h/:/;

$c = ":";
@heures = qw(08h45 09h34 13h45);
s/h/$c/o for @heures;

$tab_dir[$i] =~ s/gauche/droite/;
$hh{"matin"} =~ s/h/:/;
```

Opérateur de translittération 1/2

iode conseil
ingénierie de la formation à distance

- *lvalue* =~ tr/*cars_rech*/*cars_remp*/ ou tr/*cars_rech*/*cars_remp*/ (utilise *$_*) remplace dans *lvalue* chaque caractère de *cars_rech* par le caractère correspondant dans *cars_remp* et retourne le nb. de caractères remplacés ou supprimés
- *cars_rech* et *cars_remp* ne sont pas des expressions régulières et aucune interpolation n'est réalisée
- Modificateurs : tr/*cars_rech*/*cars_remp*/cds :
  - /c : complémenter *cars_rech*
  - /d : supprimer les caractères trouvés, mais non remplacés
  - /s : concentrer les caractères dupliqués remplacés

Opérateur de translittération 2/2

iode conseil
ingénierie de la formation à distance

- Exemples :

```
tr/aeiou/%/;      # "oe" => "%%"
tr{/\\. }{_};     # "./\" => "___"
$nb = tr/0-9//; # nb de chiffres de $_

$mot =~ tr/a-zA-Z//s;
# "connaissance" => "conaisance"

tr/-%@/_/d;       # "@pos-x%" => "pos_x"

$chemin =~ tr/a-zA-Z/_/cs;
# "nb.pts--en-x" => "nb_pts_en_x"
```

Fonctions split et join 1/3

iode conseil
ingénierie de la formation à distance

- split permet de diviser une chaîne en champs, une expression régulière spécifiant les séparateurs
- Exemples 1/2 :

```
$ch = "Jacques:François:Paris";
@l = split(/:/, $ch);
# => @l = ("Jacques", "François", "Paris")
$_ = "Paul::Lyon",
($prenom, $nom, $ville) = split(/:/);
# => @l = ("Paul", "", "Lyon")
@l = split(/:+/);
# => @l = ("Paul", "Lyon")
@l = split(/(:)/, $ch);
# => @l = ("Jacques", ":", "François", ":",
#           "Paris")
```

Fonctions split et join 2/3

iode conseil
ingénierie de la formation à distance

- Exemples split 2/2 :

```
$_ = " ab fg  tq ";
@l = split(/ /);
# => @l = ("", "ab", "fg", "", "tq")
@l = split(/\s+/);
# => @l = ("", "ab", "fg", "tq")
@l = split " "; @l = split;
# => @l = ("ab", "fg", "tq")
@l = split " ", $_, 2;
# => @l = ("ab", "fg  tq")
```

Fonctions split et join 3/3

iode conseil
ingénierie de la formation à distance

- join assemble une série de valeurs en plaçant entre elles une chaîne de liaison :

```
@l = qw(05 61 62 63 64);
$ch = join(".", @l); # => "05.61.62.63.64"
print $ch,"\n";
$ch = join("::", 1, 2, 3);   # => "1::2::3"
$ch = join("::", (1, 2, 3)); # idem
@l = qw(a b c);
$ch = join(">", "", @l);  # => ">a>b>c"
$ch = join("<", @l, "");  # => "a<b<c<"
$ch = join "", @l;        # => "abc"
```