Flex pour Windows : Créer votre premier analyseur lexical

- *Outils*
- *Installation de l'environnement de travail*
- *Un premier analyseur lexical*
- *Expressions régulières*
- *Options Flex*
- *Variables définies par Flex*

Installation de l'environnement de travail

Etape 1 : Installer Cygwin (Branche Devel)

Etape 2 : Ajouter le dossier Cygwin\bin dans la variable d'environnement PATH

Un premier analyseur lexical

Nous allons créer un analyseur simple qui lit des chaînes à partir de l'entrée standard et réalise les actions suivantes :

- Affiche le message « Commande STOP active », si la chaîne en entrée est « STOP »
- Affiche le message « commande GO active si la chaîne en entrée est GO

Etape 1 : écriture du fichier de définition des règles « exemple1.l »

```
%option noyywrap
%%
STOP printf("Commande STOP activée");
GO printf("Commande GO activée");
%%
void main() {
      yylex();
}
```

Etape 2 : création du fichier lex.yy.c

Flex exemple1.l (Flex génère par défaut un analyseur lexical en c nommé lex.yy.c, pour générer un fichier avec un nom différent il faut utiliser l'option –o nomFichier.c)

Etape 3 : compilation

gcc lex.yy.c (gcc génère par défaut un programme nommé a.exe, utilisez l'option –o pour avoir un nom de programme différent)

Etape 3 : exécution

Exécuter le programme a.exe

Expressions régulières

Les expressions régulières Flex sont une extension aux expressions régulières POSIX

Les métacaractères :

Un métacaractère est un caractère spécial qui a une signification particulière dans une expression régulière.

Les métacaractères suivants sont supportés par Flex : ^ . [ ] $ ( ) * + ? | { } \

Les symboles supportés dans une expression régulière

a	le caractère a, tout caractère qui n'est pas un métacaractère représente lui-même
.	Un caractère quelconque à l'exception du caractère \n.
^	Indique le début d'une ligne.
$	Indique la fin d'une ligne
[ ]	Définit une classe de caractères, exemple [0-9] est équivalent à [0123456789] ce qui représente un caractère appartenant à l'ensemble {0 ,1,2,3,4,5,6,7,8,9}.
[f-m]	un caractère compris entre f et m
[^a-z\n]	un caractère n'appartenant pas à la classe [a-z] et différent du caractère \n
[a-z]{-} [aeiou]	Un caractère appartenant à l'ensemble
exp*	0 ou plusieurs occurrences de l'expression régulière exp
exp+	0 ou 1 occurrence de l'expression régulière exp
exp{2,5}	Une concaténation de deux à cinq occurrences de l'expression exp
exp{2,}	au minimum une concaténation de deux occurrences de l'expression exp
exp{4}	exactement quatre occurrences de l'expression exp
|	Définit une alternative
()	Permet de grouper des expressions régulières
{nom_expression}	Une expression nommée
"…"	Le contenu entre " est traité littéralement
\x	Une séquence d'échappement ANSI-C si x est un 'a', 'b', 'f', 'n', 'r', 't', ou 'v', le caracètre \ est aussi utilisé pour échapper un métacaractère, par exemple pour avoir le caractère ^ dans une expression régulière il faut le faire précéder du caractère \).
exp1/exp2	exp1 mais seulement si elle est suivie de **exp2**.

Options Flex :

- Options en ligne de commande
  - –oNomFichier : l'analyse lexicale sera générée dans un fichier nommé NomFichier
  - -d : exécution en mode débogage.
  - –i : génère un analyseur lexical insensible à la casse.

  - -+ : l'analyseur est généré en c++.
- Options dans le fichier de description de l'analyseur.
  - caseless ou case-insensitive : équivalente à l'option –i
  - yylineno : Flex maintient le numéro de la ligne en cours dans la variable yylineno.
  - noyywrap: la fonction noyywrap() n'est pas appelée dans l'analyseur lexical généré.

Variables définies par Flex

- char * yytext: contient l'élément lexical (lexème ou token) reconnu par Flex .
- int yyleng : nombre de caractères du lexème courant.
- File* yyin : fichier en entrée.
- File * yyout : fichier en sortie.