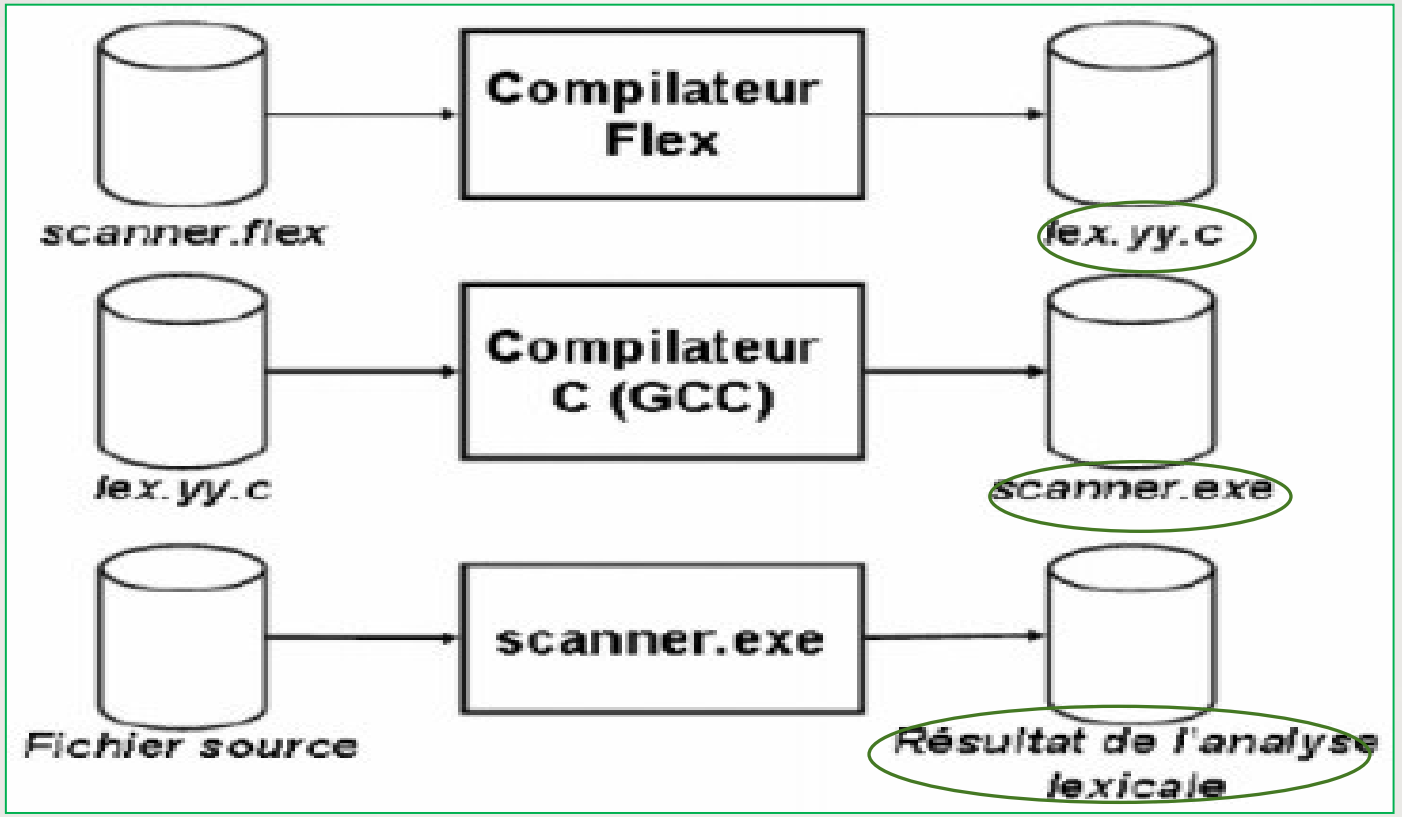
Outil Flex

Description du l'outil Flex

- **Flex** (Fast Lexer) est un compilateur pour **la génération automatique d'analyseurs lexicaux**.
- Un fichier Flex est un fichier texte qui contient la description d'un analyseur lexical en termes d'expressions régulières et d'actions écrites en langage *C*.
- Le compilateur Flex prend en entrée un fichier source Flex et produit en sortie un fichier contenant le code *C* (ou *C* ++) du future analyseur lexical.
- Le fichier *C* généré est nommé « **lex.yy.c** ».
- Ce dernier doit être compilé à l'aide d'un compilateur *C* (gcc, par exemple) pour obtenir le code exécutable de l'analyseur lexical.
- Un fichier Flex porte l'extension « **.lex** » ou «**.flex** »

Description du l'outil Flex



Étapes de développement d'un analyseur lexical avec Flex

Description du l'outil Flex

- Une fois l'analyseur lexical est mis en oeuvre, il analyse le fichier source pour chercher les occurrences d'expressions régulières.
- Lorsqu'un mot est reconnu, l'analyseur lexical exécute le code *C* correspondant à l'expression régulière qui dénote ce mot.

Procédure

- On crée le fichier source avec un éditeur de texte.
- On compile le fichier source par : **flex scan.flex**
- On compile le fichier obtenu "lex.yy.c" par : **gcc -o scan lex.yy.c –lfl**
- Lancer l'analyseur en utilisant le nom de celui ci : **./scan**

Description du l'outil Flex

- Une fois l'analyseur lexical est mis en oeuvre, il analyse le fichier source pour chercher les occurrences d'expressions régulières.
- Lorsqu'un mot est reconnu, l'analyseur lexical exécute le code *C* correspondant à l'expression régulière qui dénote ce mot.

Procédure

- On crée le fichier source avec un éditeur de texte.
- On compile le fichier source par : **flex scan.flex**
- On compile le fichier obtenu "lex.yy.c" par : **gcc -o scan lex.yy.c –lfl**
- Lancer l'analyseur en utilisant le nom de celui ci : **./scan**

Description du l'outil Flex

- Par défaut, le texte à analyser est lu à partir de l'entrée standard (le clavier) et le résultat de l'analyse lexical est affichée sur la sortie standard (l'écran).
- On peut indiquer au compilateur le fichier source à analyser : **scan <input.txt**
- On peut indiquer au compilateur d'afficher le résultat d'analyse dans un fichier : **scan>output.txt**
- Bien sur on peut mélanger les deux :

  scan<input.txt>output.txt

Format d'un programme Flex

- Le contenu d'un fichier Flex comprend 3 sections séparées par une ligne contenant le symbole « %% » :

Section des définitions

%%

Section des règles

%%

Section du code complémentaire

Format d'un programme Flex

Section des définitions :

- Contient la déclaration des variables et des fonctions globales, des inclusions de fichiers et des expressions régulières.
- Les inclusions de fichiers et les déclarations des variables et des fonctions globales sont mises entre les symboles « %{ » et « %} » (chacun sur une seule ligne qui ne doit pas être indentée).

Format d'un programme Flex

Section des définitions :

Exemple :

```
%{
#include<stdio.h>
#define N 100
int x = 12;
float z = 1.5; // une variable globale
int fct(int, float); // une fonction globale
%}
```

Format d'un programme Flex

Définition d'une expression régulière :

- Syntaxe :

  Nom Expression_Régulière

- « Nom » : un identificateur qui sert à nommer une expression régulière pour l'avoir référencer ultérieurement.
- Le nom doit être non indenté.
- Une définition d'une expression régulière doit tenir sur une seule ligne.

Format d'un programme Flex

Définition d'une expression régulière :

- Exemple :

  L [a-zA-Z]

  D [0-9]

  U "_"

  Ident ({L} | {U}) ({L} | {D}| {U}) *

Format d'un programme Flex

Section des règles :

- Une **règle** se présente sous la forme :

  Motif Action

- Un motif ne doit pas être indenté et les actions doivent commencer dans la même ligne que leur motif.
- Un motif est une expression régulière qui peut référencer les expressions régulières définies dans la section des définitions.

Format d’un programme Flex

Section des règles :

- Une action est séparée de son motif par au moins un espace (ou une tabulation).
- L’analyseur lexical déclenche une action autant de fois qu’il trouve un lexème qui correspond au motif associée à cette action.
- Les règles sont toutes écrites entre les deux symboles « %% ».

Format d'un programme Flex

Exemple:

```
Digit       [0-9]
%%
{Digit} + printf("Un nombre\n");
"Flex"  printf("Flex\n");
%%
```

Format d'un programme Flex

Section du code complémentaire:

- Sera **copiée telle quelle** dans le fichier «lex.yy.c ».
- Contient des **routines** pouvant être appelées par l'analyseur lexical.
- Peut aussi contenir une fonction « main ».
- Section optionnelle : si elle est absente, le second symbole « %% » peut être omis.

Format d'un programme Flex

Exemple complet:

```
%{
#define N 10
int tab[N];
int i = 0;
int add(int[], int);
%}
nombre    [0-9]+
```

Format d'un programme Flex

Exemple complet:

```
%%
{nombre} tab[i++] = atoi(yytext);
.        ;
%%
```

Format d'un programme Flex

Exemple complet:

```
int add(int a[], int n)
{
   int s = 0, i;
   for(i = 0; i < n; i++)  s += a[i];
   return s;
}

main()
{
   yylex();
   printf("Somme de tous les nombres : %d\n", add(tab, i));
}
```

Structure d'un fichier Flex

Exemple:

. **Écrire un analyseur lexical qui remplace toute occurrence de lettre 'a' par une lettre 'z' et inversement.**

Solution:

```
%%
''a'' printf(''z'') ;
''z'' printf(''a'') ;
```

Format d'un programme Flex

Fonctionnement:

- Les deux symboles %% marquent le début de la section des règles.
- Ils doivent être écrits en première colonne.

- Le code : ''a'' printf(''z'') ; est une règle.
- Chaque règle contient deux éléments : **1. Une ExpReg**

2. Du code C

Format d'un programme Flex

Remarques:

- Une ExpReg doit toujours commencer en première colonne.
- Il faut laisser au moins un caractère blanc entre l'ExpReg et le code C associé.
- Le code C peut contenir plusieurs instructions. Les accolades {et} doivent être utilisés si c'est le cas.

Structure d'un fichier Flex

Exemple:

```
%{
    %}
    %%
    ''a'' printf(''Lecture de la lettre a'') ;
    %%
    main()
    {
    yylex() ;
    }
```

. La fonction yylex est une routine qui effectue l'appel explicite de l'analyseur lexical.

Format d'un programme Flex

- Une règle Flex est la donnée **:**

  1. d'une expression régulière.

  2. d'une action (celle qui sera exécuté lorsqu'une séquence d'entrée est reconnue).

- Exemple :

  ''int'' {printf(''Mot clé int'');}

- A chaque fois que la chaîne de caractère 'int' sera reconnue, le message ''Mot clé int'' sera afch

  sur la sortie standard.

Règles

- Flex utilise les ExpReg étendues.
- Dans Flex, les ExpReg sont construites à partir des caractères et d'opérateurs.
- Les opérateurs de Flex sont :

  " \ [] ^ - ? . * + | () $ {} % <>

Règles

Spécifications d'expressions régulières avec Flex

- '' '' : la chaîne elle même.

''abc'' spécifie la chaîne abc.

- ' ' : caractère simple.

'x' : reconnaît le caractère x.

- [] : un élément de l'ensemble.

[abc] : a, b ou c

[a-z] : toutes les lettres minuscules

- . : tout caractères sauf \n.
- * : 0 ou plusieurs occurrences.

(x|y) * : 0 ou plusieurs caractères x ou y.

Règles

Spécifications d'expressions régulières avec Flex

- **+ : une ou plusieurs occurrences.**

[a-z]+ : 1 ou plusieurs lettres minuscules.

- **| : l'alternance**

(ab|bc) : la chaîne ab ou la chaîne bc.

- **? : opérateur d'occurrence 0 ou 1 fois.**

AB?C : la chaîne ABC ou AC.

- **/ : condition de reconnaissance.**

(ab/cd) : la chaîne ab seulement si elle est suivie de la chaîne cd.

- **$: reconnaissance de fin de ligne.**

Ab$ la chaîne Ab seulement si elle est en fn de ligne.