Compilation-ISTIC

Introduction à Flex

Amira BELHEDI
belhedi.amira@yahoo.fr

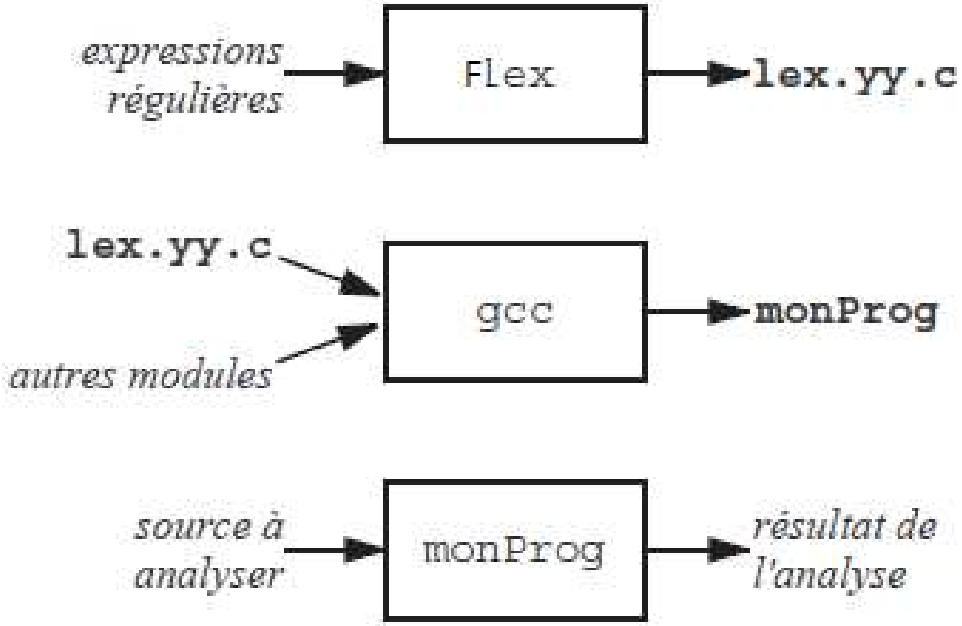
Référence

Livre: Romain Legendre, François schwarzentruber, compilation analyse lexicale et syntaxique
support de cours: Alexis Nasr
support de cours: Anne BERRY

Utilisation de générateur d'analyseur lexical

- Exemple de générateur d'analyseur lexical (Flex)
  - Flex: FastLex
  - Lex: générateurs d'analyseurs lexicaux/syntaxiques en C 'années 1970, laboratoires Bell
- il prend en entrée un ensemble d'expressions régulières
- produit en sortie le texte source d'un programme C qui, une fois compilé, est l'analyseur lexical correspondant au langage défini par les expressions régulières en question.

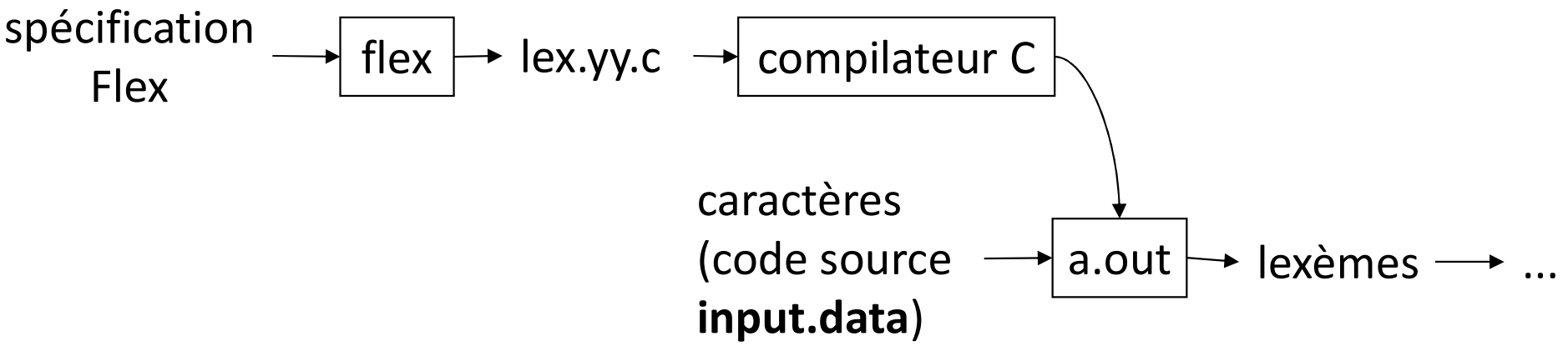
Utilisation de Flex



• Flex **produit** un fichier source **C**, nommé **lex.yy.c**, contenant la définition de la fonction int yytext(void) qui revoie un nouveau lexème

•un programme appelle cette fonction et elle renvoie une unité lexicale reconnue dans le texte source.

Utilisation de Flex



- Etapes de Génération d'un analyseur lexicale:
  - Créer sous éditeur une spécification Flex (fichier avec l'extension **.flex**)
  - Traiter cette spécification par la commande flex
  - Compiler le programme source C obtenu
  - Exécuter le programme exécutable obtenu

exemple:
Spec.flex

```
> flex  spec.flex

> gcc   lex.yy.c  -ll

> ./a.out  <  input.data
```

Spécifications Flex

- Un fichier source Flex, doit avoir un nom qui se termine par **.flex** et est fait de quatre sections délimitées par **%%**:

```
%{
    déclarations pour le compilateur C
%}
   définitions régulières
    %%
    règles de traduction
    %%
    fonctions C supplémentaires
```

Section obligatoire

Spécifications Flex

```
%{
    déclarations pour le compilateur C
%}
   définitions régulières
    %%
    règles de traduction
    %%
    fonctions C supplémentaires
```

- La partie **déclarations pour le compilateur C** et les symboles **%{** et **%}** n'est pas obligatoire
- Elle se compose de déclarations qui seront simplement recopiées au début du fichier produit, exemple, #include du fichier .h contenant les définitions des codes conventionnels des unités lexicales (INFEG, INF, EGAL, etc.).

Spécifications Flex

```
%{
    déclarations pour le compilateur C
%}
    définitions régulières
    %%
    règles de traduction
    %%
    fonctions C supplémentaires
```

- La partie **fonctions C supplémentaires** peut être absente également (le symbole **%%** qui la sépare de la deuxième section peut alors être omis).
- Elle se compose de fonctions C qui seront simplement recopiées à la fin du fichier produit.

Spécifications Flex

- Les définition régulière de la forme
  identificateur expressionRégulière
- Exemples :

  lettre [A-Za-z]

  chiffre [0-9]
- Les identificateurs ainsi définis peuvent être utilisées dans les règles de traduction et dans les définitions suivantes ➔ Il faut alors les encadrer par des accolades
- Exemples :

  lettre [A-Za-z]

  chiffre [0-9]

  alphanum {lettre}|{chiffre}

Spécifications Flex

- Les règles de traduction sont de la forme

  p_1 { $action_1$ }
  p_2 { $action_2$ }
  ...
  p_n { $action_n$ }

- où chaque p_i est une expression régulière et chaque action une suite d'instructions en C.
- Si plusieurs instructions dans la section action ➔ il faut les encadrer entre accolades {}
- Exemples

```
if    { printf ("KW : SI");  }
{lettre}{alphanum}*     return IDENTIF;
```

Spécifications Flex

- La compilation d'une source flex produit une fonction **yylex()**
- Un appel de **yylex()** déclenche une analyse lexicale du flux d'entrée
- Quand une unité lexicale est reconnue
  - le lexème reconnu est chargé automatiquement dans la variable **yytext**, de type chaine de caractères, exemple:

    (+|-)?[0-9]+ { yylval = atoi(**yytext**); return NOMBRE; }

    ➔ cette règle reconnait les nombres entiers et en calcule la valeur dans une variable entière **yylval** :
    - (atoi() = fonction de conversion d'une chaine de caractères en un une valeur entière)
  - de plus, la longueur (nombre de caractères) du lexème reconnu est chargée dans la variable entière **yyleng**

Spécifications Flex
exemple 1

```
%{
/*Partie en langage C: définitions de constantes,
déclarations de variables globales, commentaires... */
%}
delim   [ \t\n]
letter  [a-zA-Z]
%%
{letter}({letter}|[0-9])*      {printf("ID %s ", yytext ); }

([0-9]+(\.[0-9]*)?|\.[0-9]+)((E|e)(\+|-)?[0-9]+)?
                               {printf("Number %s ", yytext ); }

{delim}*        { /* pas d'action */ }
%%
```

déclarations en C

définition régulière

règles de traduction

Spécifications Flex
exemple 2

```
%{
 #include <stdio.h>
 int total = 0;
 int score = 0;
%}
LETTRE    [a-zA-Z]
CHIFFRE   [0-9]
MOT       {LETTRE}+
NOMBRE    {CHIFFRE}+
%%
{NOMBRE}   {printf("\n nombre: %s", yytext );
            total+= atoi( yytext );}
{MOT}      printf("\n mot: %s", yytext );
  .        printf("\nNi mot, ni nombre :%s", yytext);
%%
 int main( void ) {
            yylex() ;
            printf("\nSomme des nombres %d\nbye...\n", total);
 }
```

déclarations en c

définition régulière

règles de traduction

Fonctions C supplémentaires

Expression régulière Flex

Une expression régulière Flex se compose de caractères normaux et de méta-caractères qui ont une signification spéciale, comme : **$ \ ^ [ ] { } ( ) + - * / | ?**

Opérateur	Description	Exemple
c	Tout caractère c qui n'est pas un méta- caractère	Abr 120
\c	Si c est un méta-caractère alors le caractère c littéralement	\+ \. \[Considérer + . [comme symbole
"s"	La chaîne de caractère s littéralement	"abc *+ " Chaine de caractère abc*+
r1r2	r1 suivie de r2 (concaténation)	Ab
.	N'importe quel symbole sauf \n (retour à la ligne)	ab
^	prédicat qui indique ce qui doit être en début d'une ligne, il s'applique comme condition à la totalité du reste de l'expression régulière	^ab ➔ mots commençant par ab
$	prédicat qui indique ce qui doit être en fin de ligne, il s'applique comme condition à la totalité du reste de l'expression régulière	ab$ ➔ mots se terminant par ab

Expression régulière Flex

Opérateur	Description	Exemple
[liste]	Un des caractères entre crochets	[aeiou] ➔ une voyelle
[^liste]	Un caractère n'étant pas entre crochets	[^aeiou] ➔ un symbole qui n'est pas une voyelle
r*	0 ou plusieurs occurrences de r	a*
r+	1 ou plusieurs occurrences de r	a+
r?	0 ou 1 occurrence de r	a?
r{m}	exactement m occurrences de r	a{3} aaa
r{m,n}	entre m et n occurrences de r	a{5,6} aaaaa ou aaaaaa
r1|r2	r1 ou r2	a|b
r1/r2	Reconnait r1 si elle figure avant r2	**Cours de / compilation** Ne reconnait pas ***Cours de*** dans la phrase ***Cours de TIRM***
( )	Groupement de l'expression entre parenthèses	(abc) ➔ la chaine abc
\n	Caractère « retour à la ligne »	
\t	Tabulation	
{ }	Pour faire référence à une définition régulière	{NOMBRE}
EOF	Fin de fichier (uniquement avec flex)	