Initiation à FLEX

1 Présentation

Un analyseur lexical permet de reconnaître des motifs dans une suite de caractères, puis d'exécuter une action en fonction du motif reconnu. Par exemple, le motif *if x !=3.5 then* peut être analysé de la façon suivante :

if	$\longrightarrow$	Afficher 'mot réservé'
x	$\longrightarrow$	Afficher 'identificateur'
!=	$\longrightarrow$	Afficher 'opérateur de test'
3.5	$\longrightarrow$	Afficher 'flottant'
then	$\longrightarrow$	Afficher 'mot réservé'

FLEX (Fast LEX) permet de générer des analyseurs lexicaux en C. Pour cela, FLEX lit un fichier source (par convention d'extension .l ou .lex) décrivant l'analyseur à générer. Cette description est sous la forme d'un ensemble de paires (ou règles), chacune étant constituée d'une expression rationnelle et d'un code C. FLEX génère un code source C (lex.yy.c par défaut) définissant une fonction *yylex()* analysant le texte d'entrée. Ce code source est compilé (avec la librairie FLEX : -lfl à préciser à la fin de l'instruction de compilation). L'exécutable obtenu repère dans le texte d'entrée les motifs décrits par les expressions rationnelles et pour chacun d'eux exécute le code C associé.

2 Format d'un fichier FLEX

Partie Définitions
%%
Partie Règles
%%
Partie Code utilisateur

Table 1 – Structure d'un fichier FLEX

2.1 Partie Définitions

Cette partie contient :
— *include* + déclarations de fonctions, variables et constantes globales du programme C, partie encadrée par %{ et %}.
— Définitions rationnelles (pour la simplification) de la partie règles (voir plus loin) de la forme :

Nom de la définition Expression rationnelle

2.2 Partie Règles

Cette partie définit le traitement lexical selon la forme :

Motif (expression rationnelle) action (code C)

Motifs et expressions rationnelles autorisées : *cf* liste des motifs.

Si une définition est située dans la partie correspondante (partie précédente), le motif peut être utilisé en l'encadrant par { et }.

Par exemple, pour une définition :

$$\text{CHIFFRE} \quad [0-9]$$

on peut définir une règle :

$$\{\text{CHIFFRE}\}+ \qquad \text{printf}(\text{"Un entier : \%s(\%d)\textbackslash n"}, \text{yytext}, \text{atoi}(\text{yytext}));$$

2.3 Partie Code utilisateur

Cette partie est optionnelle tout comme le %% qui la précède. Elle sera copiée telle quelle dans le programme C. Les symboles %{ et %} nécessaires pour les deux parties précédentes ne sont pas utilisés dans cette partie.

3 Compilation Basique d'un fichier FLEX

$$\text{monfichier.l} \xrightarrow{\text{flex monfichier.l}} \text{lex.yy.c} \xrightarrow{\text{gcc lex.yy.c -lfl}} \text{a.out (fichier exécutable)}$$

Un fichier `makefile` est fourni. Il contient la description des options de compilation utilisées.

4 Fonctions et variables prédéfinies

Les fonctions et variables suivantes sont une partie des fonctions déjà définies :

- *yytext* variable de type *char **, contient le dernier motif reconnu par l'analyseur.
- *yyleng* variable de type *int*, contient la longueur du dernier motif reconnu par l'analyseur.
- *yyin* et *yyout* variables de type *file **, permettent de gérer l'entrée et la sortie.
- *ECHO* affiche le contenu de *yytext*.
- *REJECT* renvoie le motif dans l'analyseur, qui tente d'appliquer une autre règle sur ce motif.
- *yyless(n)* avec *n* un entier, renvoie sur le flot d'entrée le contenu de *yytext* privé des *n* premiers caractères.
- *yymore()* concatène le prochain motif reconnu à *yytext* (au lieu de prendre sa place).
- *input()* mange le prochain caractère du flot d'entrée, et le renvoie.
- *unput(c)* insère le caractère *c* en tête du flot d'entrée.
- *yywrap()* fonction appellée lors d'une rencontre d'une fin de fichier. Elle renvoie 1 si l'analyse est terminée, 0 sinon. Dans le premier cas, *yylex* appelle la fonction *yyterminate* après avoir éventuellement

fait appel aux actions de la règle «EOF». Dans le second cas, il faut positionner la variable *yyin* sur le nouveau fichier à analyser.

5 Généralités

Dans les deux premières sections, toute portion de texte indentée ou comprise entre les motifs %{ et %} est reproduite telle quelle dans l'analyseur. On utilise donc ces motifs pour les include et les déclarations de variables et constantes de la première section (Partie Définitions). Les utiliser avant la première règle de la seconde section permet de déclarer des constantes et variables locales ou du code interne à la fonction yylex().

Si une règle peut s'appliquer sur deux motifs, le motif analysé sera le plus long.

Si deux règles peuvent s'appliquer sur un même motif, celle qui s'applique est celle qui apparaît le plus tôt dans la partie règles.

6 Rendu des TP

Chaque séance de TP donnera lieu à un compte-rendu individuel ou par binôme. Vous devrez envoyer le code correspondant aux exercices de la séance à votre chargé de TP dans les 15 jours suivants le TP. Un retard dans l'envoi, tout comme la non-résolution d'un exercice, entraînera une pénalité dans la note de TP.