Université Gustave Eiffel

Travaux Dirigés d'analyse syntaxique n°1

Licence d'informatique

———

Analyse lexicale avec `flex`

Le but de ce TD est de se familiariser avec le fonctionnement de `flex` : découpage de l'entrée, entrées-sorties, appel et retour de `yylex()`.

▸ **Exercice 1.** ***Découpage de l'entrée par un programme*** `flex`

1. *Que fait le programme* `flex` *contenu dans le fichier* `td1-ex1.lex` *?*
2. *À quoi sert la dernière règle ? Que ferait le programme si elle n'était pas là ?*
3. *À quoi sert l'avant-dernière règle ? Que ferait le programme si elle n'était pas là ?*

```
%{
/* td1-ex1.lex */
%}

%option nounput
%option noinput

%%
the   printf("English ");
of    printf("English ");
and   printf("English ");
to    printf("English ");
a     printf("English ");
his   printf("English ");
in    printf("English ");
with  printf("English ");
I     printf("English ");
which printf("English ");
de  printf("français ");
à   printf("français ");
le  printf("français ");
la  printf("français ");
et  printf("français ");
il  printf("français ");
les printf("français ");
un  printf("français ");
```

```
en  printf("français ");
du  printf("français ");
[a-zA-ZâàçêéèëîïôûùüœÂÀÇÊÉÈËÎÏÔÛÙÜŒ]+ ;
. ;
%%
```

Remarque. En UTF-8, on peut remplacer l'avant-dernière règle par :

```
([a-zA-Z]|\xc3[\x80-\xbf]|\xC5[\x92-\x93])+ ;
```

▸ Exercice 2. *Entrées-sorties*

1. *Modifiez le programme* `flex` *de l'exercice précédent :*
   — *pour qu'il compte les ocurrences de mots français et anglais au fur et à mesure qu'il les reconnait, et*
   — *pour qu'il affiche à la fin si le texte a l'air plutôt en français ou plutôt en anglais.*
   Testez en utilisant le `makefile` *qui vous est fourni.*
2. *Sauvegardez un petit texte dans un fichier et testez votre programme dessus.*
3. *Testez en redirigeant la sortie de votre programme vers un fichier résultat. Vérifiez que la sortie ne contient que le résultat demandé, sans lignes vides. Si vous corrigez, retestez.*
4. *Vérifiez que votre programme contient au moins un commentaire dans chacune des trois parties : déclarations, règles, fonctions C. Si vous ajoutez des commentaires, retestez.*

▸ Exercice 3. *La fonction* `yylex()`

Complétez le `main()` *du programme* `flex` *dans le fichier* `td1-ex3.lex` *pour qu'il chiffre un message en remplaçant des lettres par d'autres : par exemple,* pari *devient* mens, flo *devient* fut.

```
%{
/* td1-ex3.lex */
/* Chiffrement par substitution */
%}
%option nounput
%option noinput
%%
a return 'e';
c return 'd';
d return 'c';
e return 'a';
é return 'g';
g return 'e';
i return 's';
```

```
l return 'u';
m return 'p';
n return 'r';
o return 't';
p return 'm';
r return 'n';
s return 'i';
t return 'o';
u return 'l';
A return 'E';
C return 'D';
D return 'C';
E return 'A';
É return 'G';
G return 'E';
I return 'S';
L return 'U';
M return 'P';
N return 'R';
O return 'T';
P return 'M';
R return 'N';
S return 'I';
T return 'O';
U return 'L';
<<EOF>> return 0;
.|\n return yytext[0];
%%
int main() {


}
```

▸ **Exercice 4.** ***Mots d'au moins 5 lettres***

Écrire un programme `flex` *qui copie les mots d'au moins 5 lettres, sans utiliser la fonction* `strlen()` *ni la variable* `yyleng`*.*

▸ **Exercice 5.** ***Chiffrement par décalage***

Écrire un programme `flex`*, avec seulement deux règles, qui remplace dans un texte chaque lettre par sa suivante en conservant la casse (*`a` *par* `b`*,* `B` *par* `C`*,* `z` *par* `a`*). Ne pas modifier les lettres accentuées. Par exemple,* "utiliser SEULEMENT DEUX règles" *devient* "vujmjtfs TFVMFNFOU EFVY sèhmft"*.*