Algorithmes Multiplicatifs et Dichotomie.

TP-5, Module I41, Licence Informatique
université du sud Toulon-Var

Mai 2007

```
 1   #include <stdio.h>
 2   #include <stdlib.h>
 3
 4   int  main( int argc, char * argv[])
 5      {
 6      FILE * src;
 7      int car;
 8
 9      if ( argc > 1 )
10          src = fopen( argv[1], "r" );
11     else
12          src = stdin;
13
14     while ( ! feof( src ) ) {
15         car = fgetc( src );
16         printf("\n%c %x %d", car, car, car);
17     }
18     fclose( src );
19     return 0;
20     }
```

Figure 1: Analyse d'un flux.

Le programme ci-dessus analyse les caractères du flux `src`. Le nom du fichier est passé sur la ligne de commande. Le fichier est ouvert en lecture en ligne 10, par défaut la lecture est faite sur l'entrée standard du programme (12). Les caractères sont lus séquentiellement (15) tant que la fin de fichier

n'est pas renontrée (14). On ferme le fichier (18) avant de terminer le programme. Editez un fichier `scan.c` compilez en un exécutable `scan.x`. Lancez l'analyse du fichier `scan.c` puis celle de l'entrée standard.

```
$ gcc -Wall scan.c -o scan.x
$ ./scan.x scan.c
$ ./scan.x
```

Remarque : Le caractère CTRL D code ASCII $0xff$ marque la fin d'un flux. Pour obtenir la liste complète des codes ASCII consulter la page du manuel `man ascii` .

Exercice 1 *Modifiez le programme pour donner la distribution des 256 caractères dans un fichier source. Vous utiliserez* ***gnuplot*** *pour tracer l'histogramme, sous* ***gnuplot****, pour faire une représentation graphique d'un fichier* ***data.txt****, il vous suffit d'exécuter la commande:* ***plot 'data.txt'****, des options permettent de faire des variantes, voir le manuel par la commande* ***help****. Par ailleurs, les mécanismes de redirections et de pipes, ainsi que des commandes unix comme* ***grep*** *et* ***sed*** *peuvent vous aider à construire et manipuler les fichiers de données. Par exemple, supposons que l'exécution d'une commande* ***./a.out tp-5-scan.c*** *envoie sur la sortie standard :*

```
Distribution des lettres:
A :  16
B :   3
C :  29
D :   9
E :  18
F :  11
G :   5
H :   5
I :  19
J :   0
K :   0
L :  10
M :   1
N :  18
O :   8
P :   9
Q :   0
```

```
R :  30
S :  14
T :  21
U :   5
V :   2
W :   2
X :   0
Y :   2
Z :   1
bye...
```

On pourra consruire un fichier de données, contenant un nombre par ligne, en utilisant les filtres `grep` *et* `sed` *et une redirection vers un fichier :*

`./a.out tp-5-scan.c | grep ''[0-9]'' | sed ''s/.*://'' > data.txt`

La commande `gnuplot` *vous permez de faire un graphique à partir d'un fichier texte contenant une liste de nombres. Par exemple, sous gnuplot, l'instruction:*

`plot 'data.txt' with boxes`

produit l'histogramme de la figure (2).

Exercice 2 *Modifiez le programme pour compter les lettres, les chiffres, le nombre de lignes, les espaces et caractères de ponctuation et autres. Pensez à utiliser des fonctions comme* `isdigit`*.*

Exercice 3 *L'automate de la figure (3) reconnaît les listes d'identificateurs. On se propose d'écrire un scanner* `listid [source]` *qui compte le nombre de liste d'identificateur d'un fichier source.*

(a) Utilisez la construction `enum` *pour définir un type* `alphabet` *comprenant les valeurs alpha, alphanum, ouvrir, fermer, espace, virgule* et *autre.*

(b) Ecrire une fonction `alphabet itoalpha( int car )` *qui retourne la classe d'un caractère.*

(c) Compléter l'algorithme (4).

(d) Écrire un programme `listid` *qui lit les caractère d'un fichier source, pour scaner le nombre de listes d'identificateurs.*

(e) Appliquer le programme à votre source.

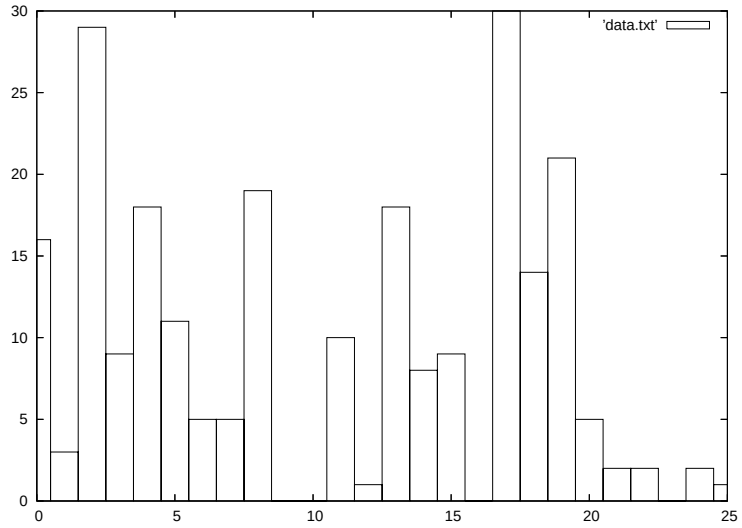

Figure 2: Un graphique construit avec `gnuplot`.

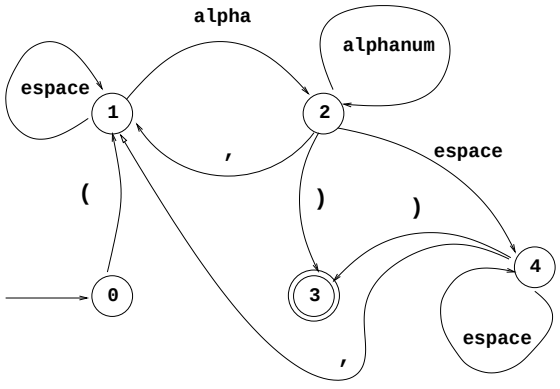

Figure 3: reconnaissance automatique d'une liste d'identificateurs.

```
LISTID( src : fichier)
VARIABLE cpt, etat, car : entier
                 cls : alphabet

DEBUT
   cpt  := 0;
   etat := 0;
   TANTQUE ( NON FINFICHIER(src)  )
     car := LIRE( src );
     cls := itoalpha( car );
     SELON ( etat )
       0 : SI ( ouvrir = cls ) ALORS etat := 1 FSI
       1 : SELON ( cls )
             espace   :
             alpha    : etat := 2;
             autre    : etat := 0;
     defaut   : etat := 0;
           FDC
       2 : SELON ( cls )
             virgule   : etat := 1;
     alpha     :
             alphanum  :
             espace    : etat := 4;
             fermer    : etat := 3;
             autre     : etat := 0;
     defaut    : etat := 0;
           FDC
       3 : SI ( ouvrir = cls ) AlORS etat := 1;
           SINON etat := 0; FSI
       4 :SELON ( cls )
             virgule   : etat := 1;
             espace    :
             fermer    : etat := 3;
             autre     : etat := 0;
     defaut    : etat := 0;
           FDC
     FDC
   FTQ
FIN
```

Figure 4: Analyse d'une source.