4.2.2 Reconnaître des mots avec lex

```
%{
/* Programme de reconnaissance très simple :
 * Verbe ou pas un verbe
 */
%}

%%

[\t ]+ /* Blancs ignorés */

suis |
es |
est |
sommes |
êtes |
sont { printf("%s est un verbe\n", yytext) ;}

[a-zA-ZÀ-ÖÙ-Üà-öù-ü]+ { printf("%s n'est pas un verbe\n", yytext) ;}

.|\n { ECHO ; /* C'est ce qui se passerait de toute façon */}

%%

main()
{
  yylex() ;
}
```

4.2.3 Compter des mots avec lex

```
%{
int charCount = 0 ; wordCount = 0 ; lineCount = 0 ;
%}

word [^ \t\n]+
eol \n
%%
{word}   { wordCount++; charCount += yyleng ; }
{eol}    { charCount++ ; lineCount++ ; }
.        charCount++ ;
%%
main()
{
  yylex() ;
  printf("%d %d %d\n",charCount, wordCount, lineCount) ;
}
```

4.2.4 Table des symboles

```
%{

/* Reconnaisseur de mots avec table des symboles
   [Levine et al. 92, O'Reilly]
*/

enum{
  LOOKUP = 0, /* Cas par défaut : recherche plutot que definition */
  VERB,
  ADJ,
  ADV,
  NOUN,
  PREP,
  PRON,
  CONJ
};

int state ;

int add_word(int type, char * word) ;
int lookup_word(char *word) ;
%}
%%
\n { state = LOOKUP ; }  /* Fin de ligne, retour à l'état par défaut */

        /* Chaque fois qu'une ligne commence par l'un des mots réservés */
        /* on commence à définir des mots de chaque type */
^verb   { state = VERB ; }
^adj    { state = ADJ ; }
^adv    { state = ADV ; }
^noun   { state = NOUN ; }
^prep   { state = PREP ; }
^pron   { state = PRON ; }
^conj   { state = CONJ ; }

[a-zA-ZÀ-ÖÙ-Üà-öù-ü]+   {
        if (state != LOOKUP)
          add_word(state, yytext) ;
        else
          switch(lookup_word(yytext)) {
            case VERB: printf("%s : verb\n", yytext) ; break ;
            case ADJ: printf("%s : adj\n", yytext) ; break ;
            case ADV: printf("%s : adv\n", yytext) ; break ;
            case NOUN: printf("%s : noun\n", yytext) ; break ;
            case PREP: printf("%s : prep\n", yytext) ; break ;
            case PRON: printf("%s : pron\n", yytext) ; break ;
            case CONJ: printf("%s : conj\n", yytext) ; break ;
            default: printf("Je n'ai pas reconnu %s\n", yytext) ;
                     break ;
          }
        }
.       /* Ignorer le reste */
```

```
%%
main()
{
  yylex() ;
}

struct word {
  char * word_name ;
  int word_type ;
  struct word * next ;
} ;

struct word * word_list ;

extern void * malloc() ;

int add_word(int type, char * word)
{
  struct word *wp;

  if(lookup_word(word) != LOOKUP) {
    printf("Problème: %s déjà défini !\n", word) ;
    return 0 ;
  }

  wp = (struct word *) malloc(sizeof(struct word)) ;

  wp->next = word_list ;

  wp->word_name = (char *) malloc(strlen(word)+1);
  strcpy(wp->word_name, word) ;
  wp->word_type = type ;
  word_list = wp ;
  return 1 ;  /* OK */
}

int lookup_word(char *word)
{
  struct word * wp = word_list;

  for ( ; wp ; wp = wp->next) {
    if (strcmp(wp->word_name, word) == 0)
       return wp->word_type ;
  }

  return LOOKUP ;  /* Pas trouvé */
}
```