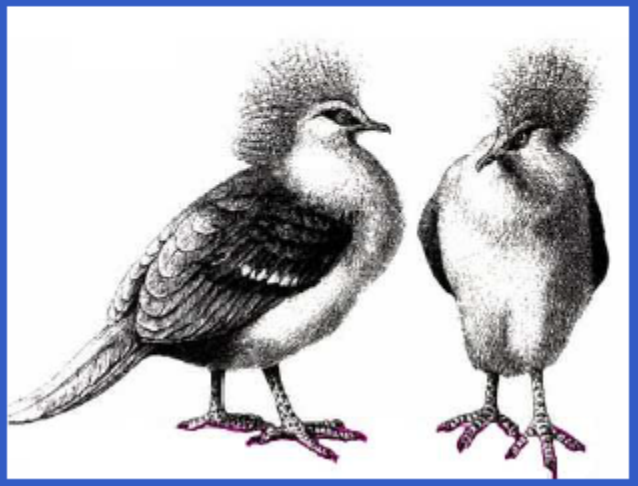
# Le générateur de *scanners* (F)LEX



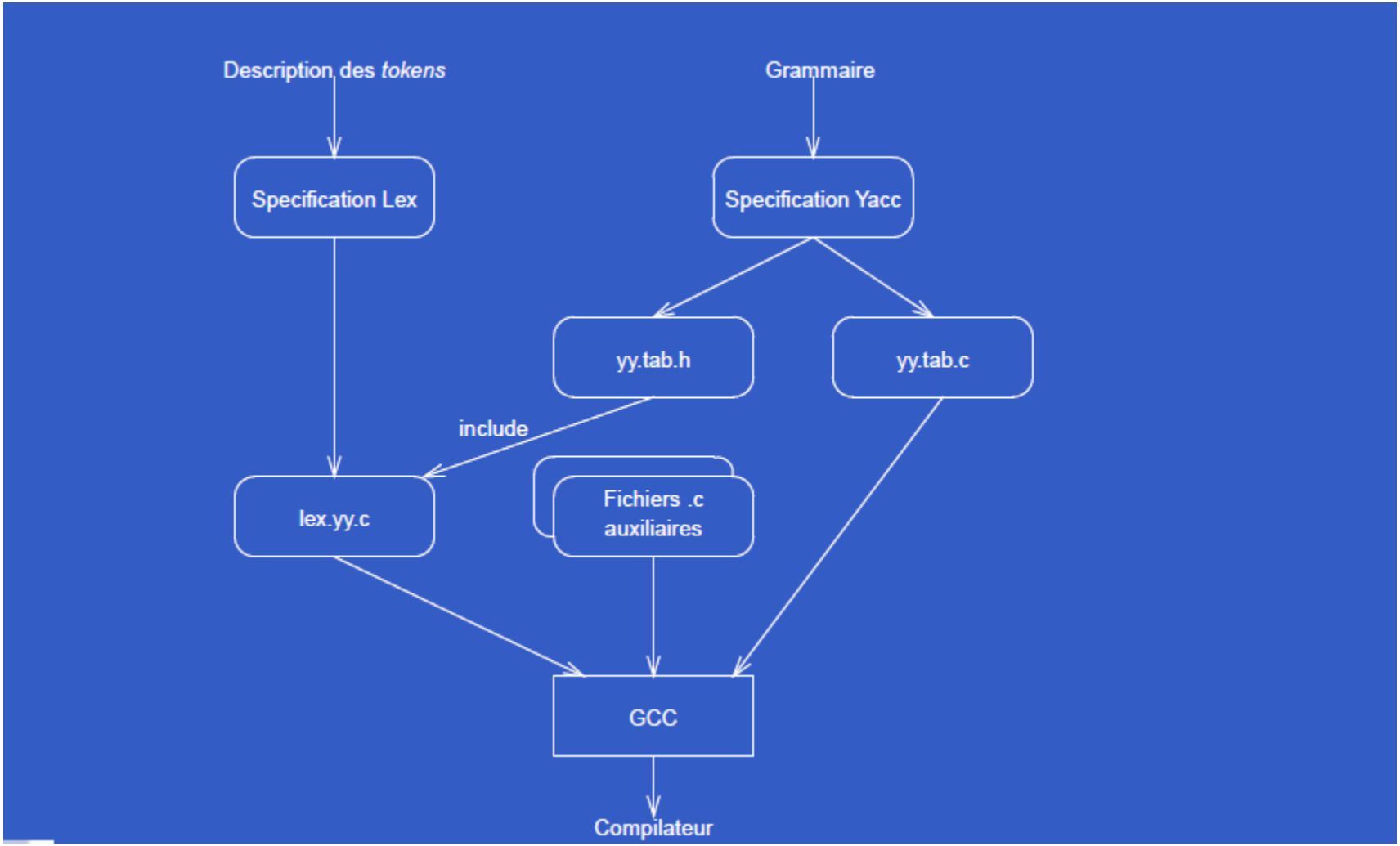
# Introduction – 1

- LEX est un outil qui permet de générer automatiquement un *scanner* à partir d'une spécification.
- Il est souvent utilisé en conjonction avec YACC (générateur d'analyseur syntaxique), qui fera l'objet d'un prochain TP. . .
- Il existe une version GNU de LEX, qui s'appelle FLEX, ainsi que de YACC, qui s'appelle BISON.

# Introduction – 2

- L'*input* de LEX est une spécification, formée de paires d'expressions régulières et de code C ;
- LEX s'en sert pour générer un fichier source C, implémentant le *scanner* sour la forme d'une fonction : `yylex()` ;
- L'exécutable, obtenu après compilation, analyse son *input* pour trouver des occurences des expressions régulières de la spécification, et exécute le code C associé.

# Interaction LEX-YACC



# Format d'une spécification LEX

- Format en trois parties, séparées par des `%%` :
  - Partie 1 : Déclaration de variables, de constantes et de définitions régulières ;
    - Les définitions régulières sont utilisées comme des « macros » dans les actions ;
    - Par exemple : `chiffre [0-9]`

# Format d'une spécification LEX

- Format en trois parties, séparées par des `%%` :
  - Partie 2 : Règles de traduction, de la forme :

    `ExpReg {Action}`

    - `ExpReg` est une expression régulière étendue ;
    - `Action` est un fragment de code C, qui sera exécuté chaque fois qu'un *token* satisfaisant `ExpReg` est rencontré.
    - Les actions peuvent faire appel aux expressions régulières de la partie 1 grâce aux { } ;
    - Par exemple : `{chiffre} {Action...};`

# Format d'une spécification LEX

- Format en trois parties, séparées par des `%%` :
  - Partie 3 : Procédures de l'utilisateur.
    - Par exemple : `main()` si le *scanner* n'est pas utilisé avec YACC ou BISON.

# Variables spéciales accessibles

- Dans les actions, on peut accéder à certaines variables spéciales :
  - `yyleng` : contient la taille du *token* reconnu ;
  - `yytext` : est une variable de type `char*` qui pointe vers la chaîne de caractères reconnue par l'expression régulière.
  - `yylval` : qui permet de passer des valeurs entières à YACC...
- Il existe aussi une action spéciale : `ECHO` qui équivaut à `printf("%s",yytext)`.

# Exemple

```
chiffre [0-9]
lettre [a-z]|[A-Z]
%%
{lettre}({chiffre}|{lettre})*   {
        printf("J'ai reconnu l'identifiant:
        %s de longueur %d\n", yytext, yyleng) ;
        }

({chiffre})+    {
        printf("J'ai reconnu un nombre\n") ; ECHO ;
        }
%%
main(){ yylex() ; }
```