NAME

flex++ – generate a scanner in c or c++..

SYNOPSIS

scanner++ [**–bcFfdIiLpsTtv?**] [**–C**[**e**][**mf**|**F**]] [**–a***tmp–directory*] [**–o***outfile*] [**–g***include–pathname*] [**–h**[*headerfile*]] [**–S***skeleton*] [**–H***header–skeleton*] *grammar–file...*

DESCRIPTION

Generate a scanner. Based on **flex** version 2.3.7. See **flex**(1) and **flexdoc**(1) for details of main functionality. Only changes are reported here.

You now generate a C++ class if you are compiling with a C++ compiler. A generated header can be generated, and is made from a skeleton–header. The code skeleton is also more adaptable. It permit you to modify much things only by changing the two skeletons.

In plain C, the **flex++** is compatible with standard **flex**.

If no header is generated, it is in fact merged, instead of included.

OPTIONS

–a *tmp–directory*
Set directory of temp files .

–S*skeleton*
Set filename of code skeleton. Default is **flexskel.cc**.

–H*header–skeleton*
Set filename of header skeleton. Default is **flexskel.h**.

–h[*header*]
Set filename of header skeleton. Default is **lex.yy.h**, or *c_basename*.h if **–o**is used and there is no header name, **.c**, **.cc**, **.C**, **.cpp**, **.cxx**... options for output files are replaced by **.h** to generate the header name.

–g*includefilename*
change the filename that **flex++** put in the **#include** inside the code, when a separate header is generated. Useful when the parameter name of the header contain pathname information that may change.

DECLARATIONS

These are new declarations to put in the declaration section :

%name *scanner_name*
Declare the name of this scanner. User for C++ class name, and to render many names unique. default is **lex**. Must be given before **%define**, or never.

%define *define_name content...*
Declare a macro symbol in header and code. The name of the symbol is **YY**_'*scanner_name*'_'*define_name*'. The content if given after, as with #define. Newline can be escaped as with #define. Many symbols are proposed for customisation.

%header{
Like **%{**, but include this text in the header. End with **%}**. When put in declaration section, the text is added before the definitions. It can be put at the begin of the second section so that the text is added after all definition in the header.

DECLARATION DEFINE SYMBOLS

These are the symbols you can define with **%define** in declaration section, or that are already defined. Remind that they are replaced by a preprocessor **#define YY**_'*scanner_name*'_'*name*.

FLEX_SCANNER
Automaticaly defined in the code. used for conditioanl code. it is effectively defined at the point of the **%name** directive, or at the point of the **%%** between section 1 and 2.

CHAR Automaticaly defined in the code. Define the type of char used depending of the 8–bits flag (**unsigned char** if 8–bit, **char** if 7–bit). it is effectively defined at the point of the **%name** directive, or at the point of the **%%** between section 1 and 2. You cannot use it before.

FLEX_DEBUG
Automaticaly defined in the code if debug option **–d** set. Define the type of char used depending of the 8–bits flag (**unsigned char** if 8–bit, **char** if 7–bit). it is effectively defined at the point of the **%name** directive, or at the point of the **%%** between section 1 and 2. You cannot use it before.

DEBUG_FLAG
The runtime debug flag name. Default is **yy_flex_debug**. See **yy_flex_debug** in flex. Used only in debug mode.

DEBUG_INIT
The runtime debug flag initial value. Default is **1**. See **yy_flex_debug** in flex.

TEXT The scanned text string. default **yytext**. See **yytext** in **flex**.

LENG The scanned text length. default **yyleng**. See **yyleng** in **flex**.

IN The input file pointer. default **yyin**. See **yyin** in **flex**.

OUT The input file pointer. default **yyout**. See **yyout** in **flex**.

LEX The scanner function name. default **yylex**. See **yylex** in **flex**. Replace **#define YYDECL**.

LEX_RETURN
The scanner function return type. default **int**. See **yylex** in **flex**. Replace **#define YYDECL**.

LEX_PARAM
The scanner function parameter list. default **void**, or empty un old–C. See **yylex** in **flex**. Replace **#define YYDECL**.

LEX_PARAM_DEF
The scanner function parameter declaration for old–C. Defined and used only in old–C. Default empty . See **yylex** in **flex**. Replace **#define YYDECL**. For example to pass an **int**, named **x**, **LEX_PARAM**

is set to **x**, and **LEX_PARAM_DEF** to **int x ;**.

RESTART
The restart function name. default **yyrestart**. See **yyrestart** in **flex**.

SWITCH_TO_BUFFER

LOAD_BUFFER_STATE

CREATE_BUFFER

DELETE_BUFFER

INIT_BUFFER
The buffer control functions names. defaults are **yy_switch_to_buffer**, **yy_load_buffer_state**, **yy_create_buffer**, **yy_delete_buffer**, **yy_init_buffer**. See this functions in **flex**.

CURRENT_BUFFER
The name of the pointeur to the current buffer. Without class, it is yy_current_buffer, and the old macro YY_CURRENT_BUFFER is defined to it's value. With class, the default value is YY_CURRENT_BUFFER, and there is no macro YY_CURRENT_BUFFER.

These are only used if class is generated.

CLASS The class name. default is the scanner name.

INHERIT
The inheritance list. Don't forget the **:** before, if not empty list.

MEMBERS
List of members to add to the class definition, before ending it.

ECHO The scanner echo member function boby. Default to yy_echo. this function is called by the macro ECHO. See **ECHO** on **flex**.

INPUT The block input member function . This function is called inside the macro YY_INPUT. It read a block of text to be scanned. Default is to read **yyin**. See **YY_INPUT**.

FATAL_ERROR
The error message member function . This function is called inside the macro YY_FATAL_ERROR. Default is to write the message to stderr and exit . See **YY_FATAL_ERROR**.

WRAP The wrap member function . This function is called inside the macro **yywrap()**. Default is to return 1 . See **yywrap()** in **flex**.

ECHO_PURE

INPUT_PURE

FATAL_ERROR_PURE

WRAP_PURE
Indicate that the corresponding member function is to be pure. It implys automatically the *function*_**NOCODE** symbol

ECHO_NOCODE

INPUT_NOCODE

FATAL_ERROR_NOCODE

WRAP_NOCODE
Indicate that the corresponding member function is not to be defined in the generated code, but outside by yourself. Activated automaticaly by the *function*_**PURE** symbols.

ECHO_CODE

INPUT_CODE

FATAL_ERROR_CODE

WRAP_CODE
Give the body code of the corresponding member function. default is to implement standard behaviour. Ignored if *function*_**PURE** or *function*_**NOCODE** are defined.

CONSTRUCTOR_PARAM
List of parameters of the constructor. Dont allows default value.

CONSTRUCTOR_INIT
List of initialisation befor constructor call. If not empty dont't forget the **:** before list of initialisation.

CONSTRUCTOR_CODE
Code added after internal initialisations in constructor.

DESTRUCTOR_CODE
Code added before internal cleanup in destructor.

IOSTREAM
If defined, this flag make flex use the **iostream** library. The behaviour is much the same, but instead of **FILE** *, yyin and yyout are **istream** * and **ostream** *. they point to **cin** and **cout** by default. Debug message and fatal error are printed on **cerr**. **BUFFER** refers to **istream** * instead of **FILE** *. These values are default, but like with **stdio** you can change them with the same **%define**. **iostream.h** is also included.

IFILE Type of the structure that represent IN file (**yyin**). Normally **FILE**, or **istream** if **IOSTREAM** is defined. **BUFFER** function use also pointer to this type.

IFILE_DEFAULT
Initial value of **IN** (yyin). Normally **stdin**, or **&cin** if **IOSTREAM** is defined.

OFILE Type of the structure that represent OUT file (**yyout**). Normally **FILE**, or **ostream** if **IOSTREAM** is defined.

OFILE_DEFAULT
Initial value of **OUT** (yyout). Normally **stdout**, or **&cout** if **IOSTREAM** is defined.

ERRFILE
File handle used to output debug message, and also fatal errors. Default is **stderr** or **cerr** if **IOSTREAM** is defined.

OBSOLETED FUNCTIONS

yyinput()
In C++, the member function **yyinput()** is equivalent to **input()** that read one char. It is kept for compatibility with old flex behaviour, that replaced in C++ ,the function **input()** with **yyinput()** not to colide with stream library. Don't mismatch it with **yy_input(char *buf, int &result, int max_size)** which read a bloc to be buffered.

OBSOLETED PREPROCESSOR SYMBOLS

if you use new features, the folowing symbols should not be used, though they are proposed. Incoherence may arise if they are defined simultaneously with the new symbol.

YYDECL
In C only. Prefer **%define LEX**, **%define LEX_RETURN**, **%define LEX_PARAM**, **%define LEX_PARAM_DEF**. Totaly ignored with classes, or if you **%define** one of these symbols, or the symbol **LEX_DEFINED**, since it mean you use the new ways to redefine yylex declaration. Never use it if header are generated, since the declared function would be wrong.

yy_new_buffer
In C only. Prefer **%define CREATE_BUFFER**.

YY_CHAR
like with old **flex**. You should better use the **%define**'ed symbol **CHAR**, or not use this yourself, since you know if you are 8 or 7–bit. Not defined in separate header.

FLEX_DEBUG
Like with old **flex**. activate trace. prefer the automaticaly added **%define DEBUG** . Defined if debug option **–d** set.

FLEX_SCANNER
like with old **flex**. defined in the scanner itself .

YY_END_TOK
Like with old **flex**. Indicate the value returned at end by yylex. Don't redefine it, since it is only informative. Value is 0.

CONSERVED PREPROCESSOR SYMBOLS

These symbols are kept, and cannot be defined elsewhere, since they control private parameters of the generated parser, or are actually unused. You can **#define** them to the value you need, or indirectly to the name of a **%define** generated symbol if you want to be clean.

YY_READ_BUF_SIZE
Size of read buffer (8192). You must undefine it to redefine it after, like like with old **flex**.

YY_BUF_SIZE
Total size of read buffer (YY_READ_BUF_SIZE *2). You must undefine it to redefine it after, except if defined by **cpp** , like with old **flex**.

yyterminate()
like with old **flex**. default return YY_NULL, that is 0.

YY_BREAK
Like with old **flex**. Don't use it, it is supported but dangerous.

YY_NEW_FILE
Action to continue scanning with the repopened file in yyin. like with old flex. Normally nor to be changed.

These are used only without classes, and you should redefine corresponding virtual function with classes, instead of the macros themselves.

ECHO like with old **flex**. With classes it is mapped to the virtual function yy_echo(), and you should not modify the macro itself. This name can be changed with **%define ECHO**.

YY_INPUT
like with old **flex**. With classes it use the virtual function yy_input(), and you should not modify the macro itself. This name can be changed with **%define INPUT**.

YY_FATAL_ERROR
like with old **flex**. With classes it is mapped to the virtual function yy_fatal_error(), and you should not modify the macro itself. This name can be changed with **%define FATAL_ERROR**.

yywrap like with old **flex**. With classes it is mapped to the virtual function yy_wrap(), and you should not modify the macro itself. This name can be changed with **%define WRAP**.

OTHER ADDED PREPROCESSOR SYMBOLS

YY_USE_CLASS
indicate that class will be produced. Default if C++.

C++ CLASS GENERATED

To simplify the notation, we note **%SYMBOLNAME** the preprocessor symbol generated with a **%define** of this name. In fact see the use of **%define** for it's real name.

Note that there is sometime symbols that differ from only an underscore _, like **yywrap** and **yy_wrap**. They are much different. In this case **yy_wrap()** is a virtual member function, and **yywrap()** is a macro.

General Class declaration

```
// Here is the declaration made in the header

class %CLASS %INHERIT

{

private:/* data */

// Secret, don't use.

private: /* functions */

void yy_initialize();

int input();

int yyinput() {return input();};

void yyunput( %CHAR c, %CHAR *buf_ptr );

// Others are secret, don't use.

protected:/* non virtual */

YY_BUFFER_STATE %CURRENT_BUFFER;
```

```
void %RESTART ( FILE *input_file );
void %SWITCH_TO_BUFFER( YY_BUFFER_STATE new_buffer );
void %LOAD_BUFFER_STATE( void );
YY_BUFFER_STATE %CREATE_BUFFER( FILE *file, int size );
void %DELETE_BUFFER( YY_BUFFER_STATE b );
void %INIT_BUFFER( YY_BUFFER_STATE b, FILE *file );
protected: /* virtual */
// these 4 virtual function may be declared PURE (=0), with the symbols like %ECHO_PURE,...
// these 4 virtual function may not be defined in the generated code, with the symbol like %ECHO_NOCODE,...
// these 4 virtual function may be defined with another code, with the symbol like ECHO_CODE,...
virtual void %ECHO();
virtual int %INPUT(char *buf,int &result,int max_size);
virtual void %FATAL_ERROR(char *msg);
virtual int %WRAP();
public:
%CHAR *%TEXT;
int %LENG;
FILE *%IN, *%OUT;
%LEX_RETURN %LEX ( %LEX_PARAM);
%CLASS(%CONSTRUCTOR_PARAM) ;
~%CLASS() ;
#if %DEBUG != 0
int %DEBUG_FLAG;
#endif
public: /* added members */
%MEMBERS
};
// this is the code for the virtual function
// may be disabled with symbol like ECHO_PURE or ECHO_NOCODE
void %CLASS::%ECHO() // echo the current token
{%ECHO_CODE}
int %CLASS::%INPUT(char * buffer,int &result,int max_size) // read a bloc of text
{%INPUT_CODE}
void %CLASS::%FATAL_ERROR(char *msg) // print a fatal error
{%FATAL_ERROR_CODE}
int %CLASS::%WRAP() // decide if we must stop input, or continue
{%WRAP_CODE}
```

Default Class declaration

```
// Here is the default declaration made in the header when you %define nothing
class lexer
{
private:/* data */
// Secret, don't use.
private: /* functions */
void yy_initialize();
int input();
int yyinput() {return input();};
void yyunput( unsigned char c, unsigned char *buf_ptr );
// Others are secret, don't use.
protected:/* non virtual */
YY_BUFFER_STATE YY_CURRENT_BUFFER;
void yyrestart ( FILE *input_file );
void yy_switch_to_buffer( YY_BUFFER_STATE new_buffer );
void yy_load_buffer_state( void );
YY_BUFFER_STATE yy_create_buffer( FILE *file, int size );
void yy_delete_buffer( YY_BUFFER_STATE b );
void yy_init_buffer( YY_BUFFER_STATE b, FILE *file );
protected: /* virtual */
virtual void yy_echo();
virtual int yy_input(char *buf,int &result,int max_size);
virtual void yy_fatal_error(char *msg);
virtual int yy_wrap();
public:
unsigned char *yytext;
int yyleng;
FILE *yyin, *yyout;
int yylex ( void);
lexer() ;
~lexer() ;
#if YY_lexer_DEBUG != 0
int yy_flex_debug;
#endif
public: /* added members */
};
// this is the code for the virtual function
```

```
void lexer::yy_echo() // echo the current token
{fwrite( (char *) yytext, yyleng, 1, yyout );}
int lexer::yy_input(char * buffer,int &result,int max_size) // read a bloc of text
{return result= fread( buffer, 1,max_size, yyin );}
void lexer::yy_fatal_error(char *msg) // print a fatal error
{fputs( msg, stderr );putc( 'n', stderr );exit( 1 );}
int lexer::yy_wrap() // decide if we must stop input, or continue
{return 1;}
```

USAGE

Should replace **flex**, because it generate a far more customisable parser, with header, still beeing compatible.

You should always use the header facility.

Use it with **bison++** (same author).

EXEMPLES

flex++ use itself to generate it's scanner. It is full compatible with classic flex.

This man page has been produced through a parser made in C++ with this version of **flex++** and our version of **bison++** (same author).

FILES

flexskel.cc
: main skeleton.

flexskel.h
: header skeleton.

ENVIRONNEMENT

DIAGNOSTICS

SEE ALSO

flex(1),**flexdoc**(1),**bison++**(1).

DOCUMENTATION

BUGS

Tell us more !

Because **flex++** put a **#include** of the generated header in the generated code, the header is necessary, and must be reachable by **cpp**. use the **–g** option to change the pathname of this file. Problems arise when the header is generated in another directory, or is moved.

Parameters are richer than before, and nothing is removed. POSIX compliance can be enforced by not using extensions. If you want to forbide them, there is a good job for you.

The grammar file scanner now support any EndOfLine sequence (CR, LF, CRLF), event inside the same file. So dont worry if it accept files from MSDOS, MacIntosh, and UNIX, with neither any message nor any problem. This is not a bug.

The automatic **%define** symbols **FLEX_DEBUG**, **FLEX_SCANNER** and **CHAR**, are added only after the **%name** directive, or at the **%%** between section 1 and 2. You cannot use them before, neither in **%header{**, nor **%{**. A good practice is to always give a name, and to give it at first. The old **#define** symbols are still defined at top for backward compatibility.

FUTUR WORKS

tell us !

POSIX compliance. is’nt it good now ?

compatibility with **flex 2.4** ? possible ?

INSTALLATION

With this install the executable is named flex++. rename it flex if you want, because it could replace **flex**. Another good name, could ne **flex_pp** like Dos version use.

TESTS

AUTHORS

Alain Coëtmeur (coetmeur@icdc.fr), R&D department (RDT) , Informatique–CDC, France.

RESTRICTIONS

The words ’we’, and ’us’ mean the author and colleages, not GNU. We don’t have contacted GNU about this, nowaday. If you’re in GNU, we are ready to propose it to you, and you may tell us what you think about.

Based on GNU version 2.3.8 of flex. Modified by the author.