**Lex.yy.c源程序**

**/\* A lexical scanner generated by flex \*/**

**/\* Scanner skeleton version:**

**\* $Header: /home/daffy/u0/vern/flex/RCS/flex.skl,v 2.91 96/09/10 16:58:48 vern Exp $**

**\*/**

**#define FLEX\_SCANNER**

**#define YY\_FLEX\_MAJOR\_VERSION 2**

**#define YY\_FLEX\_MINOR\_VERSION 5**

**#include <stdio.h>**

**/\* cfront 1.2 defines "c\_plusplus" instead of "\_\_cplusplus" \*/**

**#ifdef c\_plusplus**

**#ifndef \_\_cplusplus**

**#define \_\_cplusplus**

**#endif**

**#endif**

**#ifdef \_\_cplusplus**

**#include <stdlib.h>**

**#include <unistd.h>**

**/\* Use prototypes in function declarations. \*/**

**#define YY\_USE\_PROTOS**

**/\* The "const" storage-class-modifier is valid. \*/**

**#define YY\_USE\_CONST**

**#else /\* ! \_\_cplusplus \*/**

**#if \_\_STDC\_\_**

**#define YY\_USE\_PROTOS**

**#define YY\_USE\_CONST**

**#endif /\* \_\_STDC\_\_ \*/**

**#endif /\* ! \_\_cplusplus \*/**

**#ifdef \_\_TURBOC\_\_**

**#pragma warn -rch**

**#pragma warn -use**

**#include <io.h>**

**#include <stdlib.h>**

**#define YY\_USE\_CONST**

**#define YY\_USE\_PROTOS**

**#endif**

**#ifdef YY\_USE\_CONST**

**#define yyconst const**

**#else**

**#define yyconst**

**#endif**

**#ifdef YY\_USE\_PROTOS**

**#define YY\_PROTO(proto) proto**

**#else**

**#define YY\_PROTO(proto) ()**

**#endif**

**/\* Returned upon end-of-file. \*/**

**#define YY\_NULL 0**

**/\* Promotes a possibly negative, possibly signed char to an unsigned**

**\* integer for use as an array index. If the signed char is negative,**

**\* we want to instead treat it as an 8-bit unsigned char, hence the**

**\* double cast.**

**\*/**

**#define YY\_SC\_TO\_UI(c) ((unsigned int) (unsigned char) c)**

**/\* Enter a start condition. This macro really ought to take a parameter,**

**\* but we do it the disgusting crufty way forced on us by the ()-less**

**\* definition of BEGIN.**

**\*/**

**#define BEGIN yy\_start = 1 + 2 \***

**/\* Translate the current start state into a value that can be later handed**

**\* to BEGIN to return to the state. The YYSTATE alias is for lex**

**\* compatibility.**

**\*/**

**#define YY\_START ((yy\_start - 1) / 2)**

**#define YYSTATE YY\_START**

**/\* Action number for EOF rule of a given start state. \*/**

**#define YY\_STATE\_EOF(state) (YY\_END\_OF\_BUFFER + state + 1)**

**/\* Special action meaning "start processing a new file". \*/**

**#define YY\_NEW\_FILE yyrestart( yyin )**

**#define YY\_END\_OF\_BUFFER\_CHAR 0**

**/\* Size of default input buffer. \*/**

**#define YY\_BUF\_SIZE 16384**

**typedef struct yy\_buffer\_state \*YY\_BUFFER\_STATE;**

**extern int yyleng;**

**extern FILE \*yyin, \*yyout;**

**#define EOB\_ACT\_CONTINUE\_SCAN 0**

**#define EOB\_ACT\_END\_OF\_FILE 1**

**#define EOB\_ACT\_LAST\_MATCH 2**

**/\* The funky do-while in the following #define is used to turn the definition**

**\* int a single C statement (which needs a semi-colon terminator). This**

**\* avoids problems with code like:**

**\***

**\* if ( condition\_holds )**

**\* yyless( 5 );**

**\* else**

**\* do\_something\_else();**

**\***

**\* Prior to using the do-while the compiler would get upset at the**

**\* "else" because it interpreted the "if" statement as being all**

**\* done when it reached the ';' after the yyless() call.**

**\*/**

**/\* Return all but the first 'n' matched characters back to the input stream. \*/**

**#define yyless(n) \**

**do \**

**{ \**

**/\* Undo effects of setting up yytext. \*/ \**

**\*yy\_cp = yy\_hold\_char; \**

**YY\_RESTORE\_YY\_MORE\_OFFSET \**

**yy\_c\_buf\_p = yy\_cp = yy\_bp + n - YY\_MORE\_ADJ; \**

**YY\_DO\_BEFORE\_ACTION; /\* set up yytext again \*/ \**

**} \**

**while ( 0 )**

**#define unput(c) yyunput( c, yytext\_ptr )**

**/\* The following is because we cannot portably get our hands on size\_t**

**\* (without autoconf's help, which isn't available because we want**

**\* flex-generated scanners to compile on their own).**

**\*/**

**typedef unsigned int yy\_size\_t;**

**int yywrap()**

**{**

**return(1);**

**}**

**struct yy\_buffer\_state**

**{**

**FILE \*yy\_input\_file;**

**char \*yy\_ch\_buf; /\* input buffer \*/**

**char \*yy\_buf\_pos; /\* current position in input buffer \*/**

**/\* Size of input buffer in bytes, not including room for EOB**

**\* characters.**

**\*/**

**yy\_size\_t yy\_buf\_size;**

**/\* Number of characters read into yy\_ch\_buf, not including EOB**

**\* characters.**

**\*/**

**int yy\_n\_chars;**

**/\* Whether we "own" the buffer - i.e., we know we created it,**

**\* and can realloc() it to grow it, and should free() it to**

**\* delete it.**

**\*/**

**int yy\_is\_our\_buffer;**

**/\* Whether this is an "interactive" input source; if so, and**

**\* if we're using stdio for input, then we want to use getc()**

**\* instead of fread(), to make sure we stop fetching input after**

**\* each newline.**

**\*/**

**int yy\_is\_interactive;**

**/\* Whether we're considered to be at the beginning of a line.**

**\* If so, '^' rules will be active on the next match, otherwise**

**\* not.**

**\*/**

**int yy\_at\_bol;**

**/\* Whether to try to fill the input buffer when we reach the**

**\* end of it.**

**\*/**

**int yy\_fill\_buffer;**

**int yy\_buffer\_status;**

**#define YY\_BUFFER\_NEW 0**

**#define YY\_BUFFER\_NORMAL 1**

**/\* When an EOF's been seen but there's still some text to process**

**\* then we mark the buffer as YY\_EOF\_PENDING, to indicate that we**

**\* shouldn't try reading from the input source any more. We might**

**\* still have a bunch of tokens to match, though, because of**

**\* possible backing-up.**

**\***

**\* When we actually see the EOF, we change the status to "new"**

**\* (via yyrestart()), so that the user can continue scanning by**

**\* just pointing yyin at a new input file.**

**\*/**

**#define YY\_BUFFER\_EOF\_PENDING 2**

**};**

**static YY\_BUFFER\_STATE yy\_current\_buffer = 0;**

**/\* We provide macros for accessing buffer states in case in the**

**\* future we want to put the buffer states in a more general**

**\* "scanner state".**

**\*/**

**#define YY\_CURRENT\_BUFFER yy\_current\_buffer**

**/\* yy\_hold\_char holds the character lost when yytext is formed. \*/**

**static char yy\_hold\_char;**

**static int yy\_n\_chars; /\* number of characters read into yy\_ch\_buf \*/**

**int yyleng;**

**/\* Points to current character in buffer. \*/**

**static char \*yy\_c\_buf\_p = (char \*) 0;**

**static int yy\_init = 1; /\* whether we need to initialize \*/**

**static int yy\_start = 0; /\* start state number \*/**

**/\* Flag which is used to allow yywrap()'s to do buffer switches**

**\* instead of setting up a fresh yyin. A bit of a hack ...**

**\*/**

**static int yy\_did\_buffer\_switch\_on\_eof;**

**void yyrestart YY\_PROTO(( FILE \*input\_file ));**

**void yy\_switch\_to\_buffer YY\_PROTO(( YY\_BUFFER\_STATE new\_buffer ));**

**void yy\_load\_buffer\_state YY\_PROTO(( void ));**

**YY\_BUFFER\_STATE yy\_create\_buffer YY\_PROTO(( FILE \*file, int size ));**

**void yy\_delete\_buffer YY\_PROTO(( YY\_BUFFER\_STATE b ));**

**void yy\_init\_buffer YY\_PROTO(( YY\_BUFFER\_STATE b, FILE \*file ));**

**void yy\_flush\_buffer YY\_PROTO(( YY\_BUFFER\_STATE b ));**

**#define YY\_FLUSH\_BUFFER yy\_flush\_buffer( yy\_current\_buffer )**

**YY\_BUFFER\_STATE yy\_scan\_buffer YY\_PROTO(( char \*base, yy\_size\_t size ));**

**YY\_BUFFER\_STATE yy\_scan\_string YY\_PROTO(( yyconst char \*yy\_str ));**

**YY\_BUFFER\_STATE yy\_scan\_bytes YY\_PROTO(( yyconst char \*bytes, int len ));**

**static void \*yy\_flex\_alloc YY\_PROTO(( yy\_size\_t ));**

**static void \*yy\_flex\_realloc YY\_PROTO(( void \*, yy\_size\_t ));**

**static void yy\_flex\_free YY\_PROTO(( void \* ));**

**#define yy\_new\_buffer yy\_create\_buffer**

**#define yy\_set\_interactive(is\_interactive) \**

**{ \**

**if ( ! yy\_current\_buffer ) \**

**yy\_current\_buffer = yy\_create\_buffer( yyin, YY\_BUF\_SIZE ); \**

**yy\_current\_buffer->yy\_is\_interactive = is\_interactive; \**

**}**

**#define yy\_set\_bol(at\_bol) \**

**{ \**

**if ( ! yy\_current\_buffer ) \**

**yy\_current\_buffer = yy\_create\_buffer( yyin, YY\_BUF\_SIZE ); \**

**yy\_current\_buffer->yy\_at\_bol = at\_bol; \**

**}**

**#define YY\_AT\_BOL() (yy\_current\_buffer->yy\_at\_bol)**

**typedef unsigned char YY\_CHAR;**

**FILE \*yyin = (FILE \*) 0, \*yyout = (FILE \*) 0;**

**typedef int yy\_state\_type;**

**extern char \*yytext;**

**#define yytext\_ptr yytext**

**static yy\_state\_type yy\_get\_previous\_state YY\_PROTO(( void ));**

**static yy\_state\_type yy\_try\_NUL\_trans YY\_PROTO(( yy\_state\_type current\_state ));**

**static int yy\_get\_next\_buffer YY\_PROTO(( void ));**

**static void yy\_fatal\_error YY\_PROTO(( yyconst char msg[] ));**

**/\* Done after the current pattern has been matched and before the**

**\* corresponding action - sets up yytext.**

**\*/**

**#define YY\_DO\_BEFORE\_ACTION \**

**yytext\_ptr = yy\_bp; \**

**yyleng = (int) (yy\_cp - yy\_bp); \**

**yy\_hold\_char = \*yy\_cp; \**

**\*yy\_cp = '\0'; \**

**yy\_c\_buf\_p = yy\_cp;**

**#define YY\_NUM\_RULES 14**

**#define YY\_END\_OF\_BUFFER 15**

**static yyconst short int yy\_accept[140] =**

**{ 0,**

**0, 0, 15, 14, 12, 13, 11, 4, 7, 4,**

**5, 6, 5, 2, 3, 3, 3, 9, 9, 9,**

**9, 9, 9, 9, 9, 9, 9, 9, 9, 9,**

**9, 9, 9, 14, 3, 8, 0, 2, 10, 10,**

**9, 9, 9, 9, 9, 9, 9, 1, 9, 9,**

**9, 9, 9, 1, 9, 9, 9, 9, 9, 9,**

**9, 9, 9, 9, 9, 9, 4, 2, 0, 2,**

**9, 9, 9, 9, 9, 9, 9, 9, 9, 9,**

**9, 9, 9, 9, 9, 9, 9, 9, 9, 9,**

**9, 9, 9, 9, 9, 9, 9, 0, 9, 9,**

**9, 9, 9, 9, 9, 9, 9, 9, 9, 9,**

**9, 9, 9, 9, 9, 9, 9, 9, 9, 9,**

**9, 9, 9, 9, 9, 9, 9, 9, 9, 9,**

**9, 9, 9, 9, 9, 9, 9, 9, 0**

**} ;**

**static yyconst int yy\_ec[256] =**

**{ 0,**

**1, 1, 1, 1, 1, 1, 1, 1, 2, 3,**

**1, 1, 1, 1, 1, 1, 1, 1, 1, 1,**

**1, 1, 1, 1, 1, 1, 1, 1, 1, 1,**

**1, 4, 5, 6, 7, 1, 8, 9, 1, 10,**

**11, 12, 13, 14, 15, 16, 17, 18, 18, 18,**

**18, 18, 18, 18, 18, 18, 18, 1, 19, 20,**

**21, 22, 1, 1, 23, 23, 23, 23, 24, 23,**

**23, 23, 23, 23, 23, 23, 23, 23, 23, 23,**

**23, 23, 23, 23, 23, 23, 23, 23, 23, 23,**

**25, 26, 27, 1, 28, 1, 29, 30, 31, 32,**

**33, 34, 35, 36, 37, 23, 38, 39, 40, 41,**

**42, 43, 23, 44, 45, 46, 47, 48, 49, 23,**

**50, 51, 52, 53, 54, 1, 1, 1, 1, 1,**

**1, 1, 1, 1, 1, 1, 1, 1, 1, 1,**

**1, 1, 1, 1, 1, 1, 1, 1, 1, 1,**

**1, 1, 1, 1, 1, 1, 1, 1, 1, 1,**

**1, 1, 1, 1, 1, 1, 1, 1, 1, 1,**

**1, 1, 1, 1, 1, 1, 1, 1, 1, 1,**

**1, 1, 1, 1, 1, 1, 1, 1, 1, 1,**

**1, 1, 1, 1, 1, 1, 1, 1, 1, 1,**

**1, 1, 1, 1, 1, 1, 1, 1, 1, 1,**

**1, 1, 1, 1, 1, 1, 1, 1, 1, 1,**

**1, 1, 1, 1, 1, 1, 1, 1, 1, 1,**

**1, 1, 1, 1, 1, 1, 1, 1, 1, 1,**

**1, 1, 1, 1, 1, 1, 1, 1, 1, 1,**

**1, 1, 1, 1, 1**

**} ;**

**static yyconst int yy\_meta[55] =**

**{ 0,**

**1, 1, 1, 1, 1, 1, 1, 1, 1, 1,**

**1, 1, 1, 1, 1, 2, 1, 3, 1, 1,**

**1, 1, 3, 3, 1, 1, 1, 4, 3, 3,**

**3, 3, 3, 3, 3, 3, 3, 3, 3, 3,**

**3, 3, 3, 3, 3, 3, 3, 3, 3, 3,**

**3, 1, 1, 1**

**} ;**

**static yyconst short int yy\_base[142] =**

**{ 0,**

**0, 0, 207, 208, 208, 208, 208, 185, 208, 208,**

**208, 208, 193, 39, 183, 182, 181, 0, 14, 30,**

**27, 23, 26, 159, 33, 158, 170, 165, 34, 147,**

**155, 153, 158, 140, 208, 208, 174, 57, 60, 69,**

**0, 149, 157, 144, 159, 146, 152, 138, 139, 136,**

**140, 137, 134, 0, 46, 138, 141, 131, 134, 44,**

**57, 138, 131, 51, 52, 136, 208, 75, 154, 153,**

**131, 140, 135, 123, 52, 137, 135, 131, 123, 133,**

**119, 121, 124, 117, 110, 112, 114, 121, 107, 105,**

**105, 117, 107, 111, 115, 117, 106, 87, 106, 97,**

**105, 94, 101, 93, 91, 93, 90, 102, 92, 96,**

**101, 100, 98, 88, 93, 81, 93, 84, 85, 90,**

**90, 80, 88, 85, 87, 71, 80, 82, 73, 76,**

**65, 65, 77, 70, 70, 55, 57, 29, 208, 104,**

**105**

**} ;**

**static yyconst short int yy\_def[142] =**

**{ 0,**

**139, 1, 139, 139, 139, 139, 139, 139, 139, 139,**

**139, 139, 139, 140, 139, 139, 139, 141, 141, 141,**

**141, 141, 141, 141, 141, 141, 141, 141, 141, 141,**

**141, 141, 141, 139, 139, 139, 139, 140, 140, 140,**

**141, 141, 141, 141, 141, 141, 141, 141, 141, 141,**

**141, 141, 141, 141, 141, 141, 141, 141, 141, 141,**

**141, 141, 141, 141, 141, 141, 139, 139, 139, 139,**

**141, 141, 141, 141, 141, 141, 141, 141, 141, 141,**

**141, 141, 141, 141, 141, 141, 141, 141, 141, 141,**

**141, 141, 141, 141, 141, 141, 141, 139, 141, 141,**

**141, 141, 141, 141, 141, 141, 141, 141, 141, 141,**

**141, 141, 141, 141, 141, 141, 141, 141, 141, 141,**

**141, 141, 141, 141, 141, 141, 141, 141, 141, 141,**

**141, 141, 141, 141, 141, 141, 141, 141, 0, 139,**

**139**

**} ;**

**static yyconst short int yy\_nxt[263] =**

**{ 0,**

**4, 5, 6, 7, 8, 9, 9, 9, 10, 9,**

**9, 11, 11, 12, 11, 9, 13, 14, 12, 15,**

**16, 17, 18, 18, 9, 9, 9, 18, 18, 19,**

**20, 21, 22, 23, 24, 18, 25, 18, 26, 27,**

**18, 18, 18, 28, 29, 30, 31, 32, 33, 18,**

**18, 9, 34, 9, 37, 42, 38, 43, 44, 47,**

**54, 49, 40, 50, 51, 45, 54, 52, 48, 59,**

**60, 46, 37, 55, 38, 139, 82, 139, 87, 61,**

**40, 69, 62, 69, 139, 89, 70, 93, 95, 54,**

**96, 54, 68, 54, 88, 94, 100, 101, 98, 69,**

**90, 69, 138, 54, 70, 39, 39, 41, 41, 54,**

**54, 137, 136, 135, 134, 54, 54, 54, 54, 54,**

**54, 133, 54, 132, 131, 54, 130, 129, 54, 128,**

**127, 126, 125, 124, 123, 54, 122, 121, 54, 120,**

**119, 118, 54, 54, 117, 116, 54, 115, 114, 113,**

**112, 111, 110, 109, 108, 107, 106, 54, 54, 105,**

**54, 104, 54, 54, 103, 102, 54, 54, 99, 54,**

**70, 70, 97, 92, 91, 86, 85, 84, 83, 81,**

**54, 80, 79, 78, 77, 76, 75, 74, 73, 72,**

**71, 68, 67, 66, 65, 64, 63, 58, 57, 56,**

**53, 35, 35, 35, 36, 35, 139, 3, 139, 139,**

**139, 139, 139, 139, 139, 139, 139, 139, 139, 139,**

**139, 139, 139, 139, 139, 139, 139, 139, 139, 139,**

**139, 139, 139, 139, 139, 139, 139, 139, 139, 139,**

**139, 139, 139, 139, 139, 139, 139, 139, 139, 139,**

**139, 139, 139, 139, 139, 139, 139, 139, 139, 139,**

**139, 139**

**} ;**

**static yyconst short int yy\_chk[263] =**

**{ 0,**

**1, 1, 1, 1, 1, 1, 1, 1, 1, 1,**

**1, 1, 1, 1, 1, 1, 1, 1, 1, 1,**

**1, 1, 1, 1, 1, 1, 1, 1, 1, 1,**

**1, 1, 1, 1, 1, 1, 1, 1, 1, 1,**

**1, 1, 1, 1, 1, 1, 1, 1, 1, 1,**

**1, 1, 1, 1, 14, 19, 14, 19, 20, 21,**

**138, 22, 14, 22, 23, 20, 25, 23, 21, 29,**

**29, 20, 38, 25, 38, 39, 55, 39, 60, 29,**

**38, 40, 29, 40, 40, 61, 40, 64, 65, 137,**

**65, 55, 68, 136, 60, 64, 75, 75, 68, 98,**

**61, 98, 135, 134, 98, 140, 140, 141, 141, 133,**

**132, 131, 130, 129, 128, 127, 126, 125, 124, 123,**

**122, 121, 120, 119, 118, 117, 116, 115, 114, 113,**

**112, 111, 110, 109, 108, 107, 106, 105, 104, 103,**

**102, 101, 100, 99, 97, 96, 95, 94, 93, 92,**

**91, 90, 89, 88, 87, 86, 85, 84, 83, 82,**

**81, 80, 79, 78, 77, 76, 74, 73, 72, 71,**

**70, 69, 66, 63, 62, 59, 58, 57, 56, 53,**

**52, 51, 50, 49, 48, 47, 46, 45, 44, 43,**

**42, 37, 34, 33, 32, 31, 30, 28, 27, 26,**

**24, 17, 16, 15, 13, 8, 3, 139, 139, 139,**

**139, 139, 139, 139, 139, 139, 139, 139, 139, 139,**

**139, 139, 139, 139, 139, 139, 139, 139, 139, 139,**

**139, 139, 139, 139, 139, 139, 139, 139, 139, 139,**

**139, 139, 139, 139, 139, 139, 139, 139, 139, 139,**

**139, 139, 139, 139, 139, 139, 139, 139, 139, 139,**

**139, 139**

**} ;**

**static yy\_state\_type yy\_last\_accepting\_state;**

**static char \*yy\_last\_accepting\_cpos;**

**/\* The intent behind this definition is that it'll catch**

**\* any uses of REJECT which flex missed.**

**\*/**

**#define REJECT reject\_used\_but\_not\_detected**

**#define yymore() yymore\_used\_but\_not\_detected**

**#define YY\_MORE\_ADJ 0**

**#define YY\_RESTORE\_YY\_MORE\_OFFSET**

**char \*yytext;**

**#line 1 "zf.l"**

**#define INITIAL 0**

**#line 2 "zf.l"**

**#include <ctype.h>**

**#include <stdio.h>**

**#include <string.h>**

**#include <stdlib.h>**

**void comment();**

**int lineno=1;**

**int wcount=0;**

**int ccount=0;**

**int er=1,ec=1,ec1=1;**

**#line 478 "lex.yy.c"**

**/\* Macros after this point can all be overridden by user definitions in**

**\* section 1.**

**\*/**

**#ifndef YY\_SKIP\_YYWRAP**

**#ifdef \_\_cplusplus**

**extern "C" int yywrap YY\_PROTO(( void ));**

**#else**

**extern int yywrap YY\_PROTO(( void ));**

**#endif**

**#endif**

**#ifndef YY\_NO\_UNPUT**

**static void yyunput YY\_PROTO(( int c, char \*buf\_ptr ));**

**#endif**

**#ifndef yytext\_ptr**

**static void yy\_flex\_strncpy YY\_PROTO(( char \*, yyconst char \*, int ));**

**#endif**

**#ifdef YY\_NEED\_STRLEN**

**static int yy\_flex\_strlen YY\_PROTO(( yyconst char \* ));**

**#endif**

**#ifndef YY\_NO\_INPUT**

**#ifdef \_\_cplusplus**

**static int yyinput YY\_PROTO(( void ));**

**#else**

**static int input YY\_PROTO(( void ));**

**#endif**

**#endif**

**#if YY\_STACK\_USED**

**static int yy\_start\_stack\_ptr = 0;**

**static int yy\_start\_stack\_depth = 0;**

**static int \*yy\_start\_stack = 0;**

**#ifndef YY\_NO\_PUSH\_STATE**

**static void yy\_push\_state YY\_PROTO(( int new\_state ));**

**#endif**

**#ifndef YY\_NO\_POP\_STATE**

**static void yy\_pop\_state YY\_PROTO(( void ));**

**#endif**

**#ifndef YY\_NO\_TOP\_STATE**

**static int yy\_top\_state YY\_PROTO(( void ));**

**#endif**

**#else**

**#define YY\_NO\_PUSH\_STATE 1**

**#define YY\_NO\_POP\_STATE 1**

**#define YY\_NO\_TOP\_STATE 1**

**#endif**

**#ifdef YY\_MALLOC\_DECL**

**YY\_MALLOC\_DECL**

**#else**

**#if \_\_STDC\_\_**

**#ifndef \_\_cplusplus**

**#include <stdlib.h>**

**#endif**

**#else**

**/\* Just try to get by without declaring the routines. This will fail**

**\* miserably on non-ANSI systems for which sizeof(size\_t) != sizeof(int)**

**\* or sizeof(void\*) != sizeof(int).**

**\*/**

**#endif**

**#endif**

**/\* Amount of stuff to slurp up with each read. \*/**

**#ifndef YY\_READ\_BUF\_SIZE**

**#define YY\_READ\_BUF\_SIZE 8192**

**#endif**

**/\* Copy whatever the last rule matched to the standard output. \*/**

**#ifndef ECHO**

**/\* This used to be an fputs(), but since the string might contain NUL's,**

**\* we now use fwrite().**

**\*/**

**#define ECHO (void) fwrite( yytext, yyleng, 1, yyout )**

**#endif**

**/\* Gets input and stuffs it into "buf". number of characters read, or YY\_NULL,**

**\* is returned in "result".**

**\*/**

**#ifndef YY\_INPUT**

**#define YY\_INPUT(buf,result,max\_size) \**

**if ( yy\_current\_buffer->yy\_is\_interactive ) \**

**{ \**

**int c = '\*', n; \**

**for ( n = 0; n < max\_size && \**

**(c = getc( yyin )) != EOF && c != '\n'; ++n ) \**

**buf[n] = (char) c; \**

**if ( c == '\n' ) \**

**buf[n++] = (char) c; \**

**if ( c == EOF && ferror( yyin ) ) \**

**YY\_FATAL\_ERROR( "input in flex scanner failed" ); \**

**result = n; \**

**} \**

**else if ( ((result = fread( buf, 1, max\_size, yyin )) == 0) \**

**&& ferror( yyin ) ) \**

**YY\_FATAL\_ERROR( "input in flex scanner failed" );**

**#endif**

**/\* No semi-colon after return; correct usage is to write "yyterminate();" -**

**\* we don't want an extra ';' after the "return" because that will cause**

**\* some compilers to complain about unreachable statements.**

**\*/**

**#ifndef yyterminate**

**#define yyterminate() return YY\_NULL**

**#endif**

**/\* Number of entries by which start-condition stack grows. \*/**

**#ifndef YY\_START\_STACK\_INCR**

**#define YY\_START\_STACK\_INCR 25**

**#endif**

**/\* Report a fatal error. \*/**

**#ifndef YY\_FATAL\_ERROR**

**#define YY\_FATAL\_ERROR(msg) yy\_fatal\_error( msg )**

**#endif**

**/\* Default declaration of generated scanner - a define so the user can**

**\* easily add parameters.**

**\*/**

**#ifndef YY\_DECL**

**#define YY\_DECL int yylex YY\_PROTO(( void ))**

**#endif**

**/\* Code executed at the beginning of each rule, after yytext and yyleng**

**\* have been set up.**

**\*/**

**#ifndef YY\_USER\_ACTION**

**#define YY\_USER\_ACTION**

**#endif**

**/\* Code executed at the end of each rule. \*/**

**#ifndef YY\_BREAK**

**#define YY\_BREAK break;**

**#endif**

**#define YY\_RULE\_SETUP \**

**YY\_USER\_ACTION**

**YY\_DECL**

**{**

**register yy\_state\_type yy\_current\_state;**

**register char \*yy\_cp, \*yy\_bp;**

**register int yy\_act;**

**#line 20 "zf.l"**

**#line 631 "lex.yy.c"**

**if ( yy\_init )**

**{**

**yy\_init = 0;**

**#ifdef YY\_USER\_INIT**

**YY\_USER\_INIT;**

**#endif**

**if ( ! yy\_start )**

**yy\_start = 1; /\* first start state \*/**

**if ( ! yyin )**

**yyin = stdin;**

**if ( ! yyout )**

**yyout = stdout;**

**if ( ! yy\_current\_buffer )**

**yy\_current\_buffer =**

**yy\_create\_buffer( yyin, YY\_BUF\_SIZE );**

**yy\_load\_buffer\_state();**

**}**

**while ( 1 ) /\* loops until end-of-file is reached \*/**

**{**

**yy\_cp = yy\_c\_buf\_p;**

**/\* Support of yytext. \*/**

**\*yy\_cp = yy\_hold\_char;**

**/\* yy\_bp points to the position in yy\_ch\_buf of the start of**

**\* the current run.**

**\*/**

**yy\_bp = yy\_cp;**

**yy\_current\_state = yy\_start;**

**yy\_match:**

**do**

**{**

**register YY\_CHAR yy\_c = yy\_ec[YY\_SC\_TO\_UI(\*yy\_cp)];**

**if ( yy\_accept[yy\_current\_state] )**

**{**

**yy\_last\_accepting\_state = yy\_current\_state;**

**yy\_last\_accepting\_cpos = yy\_cp;**

**}**

**while ( yy\_chk[yy\_base[yy\_current\_state] + yy\_c] != yy\_current\_state )**

**{**

**yy\_current\_state = (int) yy\_def[yy\_current\_state];**

**if ( yy\_current\_state >= 140 )**

**yy\_c = yy\_meta[(unsigned int) yy\_c];**

**}**

**yy\_current\_state = yy\_nxt[yy\_base[yy\_current\_state] + (unsigned int) yy\_c];**

**++yy\_cp;**

**}**

**while ( yy\_base[yy\_current\_state] != 208 );**

**yy\_find\_action:**

**yy\_act = yy\_accept[yy\_current\_state];**

**if ( yy\_act == 0 )**

**{ /\* have to back up \*/**

**yy\_cp = yy\_last\_accepting\_cpos;**

**yy\_current\_state = yy\_last\_accepting\_state;**

**yy\_act = yy\_accept[yy\_current\_state];**

**}**

**YY\_DO\_BEFORE\_ACTION;**

**do\_action: /\* This label is used only to access EOF actions. \*/**

**switch ( yy\_act )**

**{ /\* beginning of action switch \*/**

**case 0: /\* must back up \*/**

**/\* undo the effects of YY\_DO\_BEFORE\_ACTION \*/**

**\*yy\_cp = yy\_hold\_char;**

**yy\_cp = yy\_last\_accepting\_cpos;**

**yy\_current\_state = yy\_last\_accepting\_state;**

**goto yy\_find\_action;**

**case 1:**

**YY\_RULE\_SETUP**

**#line 21 "zf.l"**

**{ec=ec+yyleng;Upper(yytext,yyleng);**

**printf("%d 行 ",lineno);**

**printf("%s reserved word\n",yytext);}**

**YY\_BREAK**

**case 2:**

**YY\_RULE\_SETUP**

**#line 24 "zf.l"**

**{ccount=ccount+yyleng;ec=ec+yyleng;wcount++;printf("%d 行 ",lineno);**

**printf("%s NUM\n",yytext);}**

**YY\_BREAK**

**case 3:**

**YY\_RULE\_SETUP**

**#line 26 "zf.l"**

**{ec=ec+yyleng;ccount=ccount+yyleng;wcount++;printf("%d 行 ",lineno);**

**printf("%s relop operator\n",yytext);}**

**YY\_BREAK**

**case 4:**

**YY\_RULE\_SETUP**

**#line 28 "zf.l"**

**{ec=ec+yyleng;ccount=ccount+yyleng;wcount++;printf("%d 行 ",lineno);**

**printf("%s logic operator\n",yytext);}**

**YY\_BREAK**

**case 5:**

**YY\_RULE\_SETUP**

**#line 30 "zf.l"**

**{ec=ec+1;ccount++;wcount++;printf("%d 行 ",lineno);**

**printf("%s arithmetic operator\n",yytext);}**

**YY\_BREAK**

**case 6:**

**YY\_RULE\_SETUP**

**#line 32 "zf.l"**

**{ec++;ccount++;printf("%d 行 ",lineno);printf("%s punctuation mark\n",yytext);}**

**YY\_BREAK**

**case 7:**

**YY\_RULE\_SETUP**

**#line 33 "zf.l"**

**{ec=ec+yyleng;ccount=ccount+yyleng;printf("%d 行 ",lineno);**

**printf("%s special symbol\n",yytext);}**

**YY\_BREAK**

**case 8:**

**YY\_RULE\_SETUP**

**#line 35 "zf.l"**

**{printf("%d 行 ",lineno);printf("/\*");comment();printf(" comment statement\n");}**

**YY\_BREAK**

**case 9:**

**YY\_RULE\_SETUP**

**#line 36 "zf.l"**

**{ccount=ccount+yyleng; wcount++;ec=ec+yyleng;printf("%d 行 ",lineno);**

**printf("%s ID\n",yytext);}**

**YY\_BREAK**

**case 10:**

**YY\_RULE\_SETUP**

**#line 38 "zf.l"**

**{ ec=ec+yyleng;wcount++;ccount=ccount+yyleng;printf("%d 行 ",lineno);er=lineno;**

**printf("error: %s\n",yytext);**

**ec1=ec-yyleng;**

**printf("line:%d,col:%d\n",er,ec1);}**

**YY\_BREAK**

**case 11:**

**YY\_RULE\_SETUP**

**#line 42 "zf.l"**

**{ec=ec+1;}**

**YY\_BREAK**

**case 12:**

**YY\_RULE\_SETUP**

**#line 43 "zf.l"**

**{ec=ec+8;}**

**YY\_BREAK**

**case 13:**

**YY\_RULE\_SETUP**

**#line 44 "zf.l"**

**{lineno++;ec=1;}**

**YY\_BREAK**

**case 14:**

**YY\_RULE\_SETUP**

**#line 45 "zf.l"**

**ECHO;**

**YY\_BREAK**

**#line 795 "lex.yy.c"**

**case YY\_STATE\_EOF(INITIAL):**

**yyterminate();**

**case YY\_END\_OF\_BUFFER:**

**{**

**/\* Amount of text matched not including the EOB char. \*/**

**int yy\_amount\_of\_matched\_text = (int) (yy\_cp - yytext\_ptr) - 1;**

**/\* Undo the effects of YY\_DO\_BEFORE\_ACTION. \*/**

**\*yy\_cp = yy\_hold\_char;**

**YY\_RESTORE\_YY\_MORE\_OFFSET**

**if ( yy\_current\_buffer->yy\_buffer\_status == YY\_BUFFER\_NEW )**

**{**

**/\* We're scanning a new file or input source. It's**

**\* possible that this happened because the user**

**\* just pointed yyin at a new source and called**

**\* yylex(). If so, then we have to assure**

**\* consistency between yy\_current\_buffer and our**

**\* globals. Here is the right place to do so, because**

**\* this is the first action (other than possibly a**

**\* back-up) that will match for the new input source.**

**\*/**

**yy\_n\_chars = yy\_current\_buffer->yy\_n\_chars;**

**yy\_current\_buffer->yy\_input\_file = yyin;**

**yy\_current\_buffer->yy\_buffer\_status = YY\_BUFFER\_NORMAL;**

**}**

**/\* Note that here we test for yy\_c\_buf\_p "<=" to the position**

**\* of the first EOB in the buffer, since yy\_c\_buf\_p will**

**\* already have been incremented past the NUL character**

**\* (since all states make transitions on EOB to the**

**\* end-of-buffer state). Contrast this with the test**

**\* in input().**

**\*/**

**if ( yy\_c\_buf\_p <= &yy\_current\_buffer->yy\_ch\_buf[yy\_n\_chars] )**

**{ /\* This was really a NUL. \*/**

**yy\_state\_type yy\_next\_state;**

**yy\_c\_buf\_p = yytext\_ptr + yy\_amount\_of\_matched\_text;**

**yy\_current\_state = yy\_get\_previous\_state();**

**/\* Okay, we're now positioned to make the NUL**

**\* transition. We couldn't have**

**\* yy\_get\_previous\_state() go ahead and do it**

**\* for us because it doesn't know how to deal**

**\* with the possibility of jamming (and we don't**

**\* want to build jamming into it because then it**

**\* will run more slowly).**

**\*/**

**yy\_next\_state = yy\_try\_NUL\_trans( yy\_current\_state );**

**yy\_bp = yytext\_ptr + YY\_MORE\_ADJ;**

**if ( yy\_next\_state )**

**{**

**/\* Consume the NUL. \*/**

**yy\_cp = ++yy\_c\_buf\_p;**

**yy\_current\_state = yy\_next\_state;**

**goto yy\_match;**

**}**

**else**

**{**

**yy\_cp = yy\_c\_buf\_p;**

**goto yy\_find\_action;**

**}**

**}**

**else switch ( yy\_get\_next\_buffer() )**

**{**

**case EOB\_ACT\_END\_OF\_FILE:**

**{**

**yy\_did\_buffer\_switch\_on\_eof = 0;**

**if ( yywrap() )**

**{**

**/\* Note: because we've taken care in**

**\* yy\_get\_next\_buffer() to have set up**

**\* yytext, we can now set up**

**\* yy\_c\_buf\_p so that if some total**

**\* hoser (like flex itself) wants to**

**\* call the scanner after we return the**

**\* YY\_NULL, it'll still work - another**

**\* YY\_NULL will get returned.**

**\*/**

**yy\_c\_buf\_p = yytext\_ptr + YY\_MORE\_ADJ;**

**yy\_act = YY\_STATE\_EOF(YY\_START);**

**goto do\_action;**

**}**

**else**

**{**

**if ( ! yy\_did\_buffer\_switch\_on\_eof )**

**YY\_NEW\_FILE;**

**}**

**break;**

**}**

**case EOB\_ACT\_CONTINUE\_SCAN:**

**yy\_c\_buf\_p =**

**yytext\_ptr + yy\_amount\_of\_matched\_text;**

**yy\_current\_state = yy\_get\_previous\_state();**

**yy\_cp = yy\_c\_buf\_p;**

**yy\_bp = yytext\_ptr + YY\_MORE\_ADJ;**

**goto yy\_match;**

**case EOB\_ACT\_LAST\_MATCH:**

**yy\_c\_buf\_p =**

**&yy\_current\_buffer->yy\_ch\_buf[yy\_n\_chars];**

**yy\_current\_state = yy\_get\_previous\_state();**

**yy\_cp = yy\_c\_buf\_p;**

**yy\_bp = yytext\_ptr + YY\_MORE\_ADJ;**

**goto yy\_find\_action;**

**}**

**break;**

**}**

**default:**

**YY\_FATAL\_ERROR(**

**"fatal flex scanner internal error--no action found" );**

**} /\* end of action switch \*/**

**} /\* end of scanning one token \*/**

**} /\* end of yylex \*/**

**/\* yy\_get\_next\_buffer - try to read in a new buffer**

**\***

**\* Returns a code representing an action:**

**\* EOB\_ACT\_LAST\_MATCH -**

**\* EOB\_ACT\_CONTINUE\_SCAN - continue scanning from current position**

**\* EOB\_ACT\_END\_OF\_FILE - end of file**

**\*/**

**static int yy\_get\_next\_buffer()**

**{**

**register char \*dest = yy\_current\_buffer->yy\_ch\_buf;**

**register char \*source = yytext\_ptr;**

**register int number\_to\_move, i;**

**int ret\_val;**

**if ( yy\_c\_buf\_p > &yy\_current\_buffer->yy\_ch\_buf[yy\_n\_chars + 1] )**

**YY\_FATAL\_ERROR(**

**"fatal flex scanner internal error--end of buffer missed" );**

**if ( yy\_current\_buffer->yy\_fill\_buffer == 0 )**

**{ /\* Don't try to fill the buffer, so this is an EOF. \*/**

**if ( yy\_c\_buf\_p - yytext\_ptr - YY\_MORE\_ADJ == 1 )**

**{**

**/\* We matched a single character, the EOB, so**

**\* treat this as a final EOF.**

**\*/**

**return EOB\_ACT\_END\_OF\_FILE;**

**}**

**else**

**{**

**/\* We matched some text prior to the EOB, first**

**\* process it.**

**\*/**

**return EOB\_ACT\_LAST\_MATCH;**

**}**

**}**

**/\* Try to read more data. \*/**

**/\* First move last chars to start of buffer. \*/**

**number\_to\_move = (int) (yy\_c\_buf\_p - yytext\_ptr) - 1;**

**for ( i = 0; i < number\_to\_move; ++i )**

**\*(dest++) = \*(source++);**

**if ( yy\_current\_buffer->yy\_buffer\_status == YY\_BUFFER\_EOF\_PENDING )**

**/\* don't do the read, it's not guaranteed to return an EOF,**

**\* just force an EOF**

**\*/**

**yy\_current\_buffer->yy\_n\_chars = yy\_n\_chars = 0;**

**else**

**{**

**int num\_to\_read =**

**yy\_current\_buffer->yy\_buf\_size - number\_to\_move - 1;**

**while ( num\_to\_read <= 0 )**

**{ /\* Not enough room in the buffer - grow it. \*/**

**#ifdef YY\_USES\_REJECT**

**YY\_FATAL\_ERROR(**

**"input buffer overflow, can't enlarge buffer because scanner uses REJECT" );**

**#else**

**/\* just a shorter name for the current buffer \*/**

**YY\_BUFFER\_STATE b = yy\_current\_buffer;**

**int yy\_c\_buf\_p\_offset =**

**(int) (yy\_c\_buf\_p - b->yy\_ch\_buf);**

**if ( b->yy\_is\_our\_buffer )**

**{**

**int new\_size = b->yy\_buf\_size \* 2;**

**if ( new\_size <= 0 )**

**b->yy\_buf\_size += b->yy\_buf\_size / 8;**

**else**

**b->yy\_buf\_size \*= 2;**

**b->yy\_ch\_buf = (char \*)**

**/\* Include room in for 2 EOB chars. \*/**

**yy\_flex\_realloc( (void \*) b->yy\_ch\_buf,**

**b->yy\_buf\_size + 2 );**

**}**

**else**

**/\* Can't grow it, we don't own it. \*/**

**b->yy\_ch\_buf = 0;**

**if ( ! b->yy\_ch\_buf )**

**YY\_FATAL\_ERROR(**

**"fatal error - scanner input buffer overflow" );**

**yy\_c\_buf\_p = &b->yy\_ch\_buf[yy\_c\_buf\_p\_offset];**

**num\_to\_read = yy\_current\_buffer->yy\_buf\_size -**

**number\_to\_move - 1;**

**#endif**

**}**

**if ( num\_to\_read > YY\_READ\_BUF\_SIZE )**

**num\_to\_read = YY\_READ\_BUF\_SIZE;**

**/\* Read in more data. \*/**

**YY\_INPUT( (&yy\_current\_buffer->yy\_ch\_buf[number\_to\_move]),**

**yy\_n\_chars, num\_to\_read );**

**yy\_current\_buffer->yy\_n\_chars = yy\_n\_chars;**

**}**

**if ( yy\_n\_chars == 0 )**

**{**

**if ( number\_to\_move == YY\_MORE\_ADJ )**

**{**

**ret\_val = EOB\_ACT\_END\_OF\_FILE;**

**yyrestart( yyin );**

**}**

**else**

**{**

**ret\_val = EOB\_ACT\_LAST\_MATCH;**

**yy\_current\_buffer->yy\_buffer\_status =**

**YY\_BUFFER\_EOF\_PENDING;**

**}**

**}**

**else**

**ret\_val = EOB\_ACT\_CONTINUE\_SCAN;**

**yy\_n\_chars += number\_to\_move;**

**yy\_current\_buffer->yy\_ch\_buf[yy\_n\_chars] = YY\_END\_OF\_BUFFER\_CHAR;**

**yy\_current\_buffer->yy\_ch\_buf[yy\_n\_chars + 1] = YY\_END\_OF\_BUFFER\_CHAR;**

**yytext\_ptr = &yy\_current\_buffer->yy\_ch\_buf[0];**

**return ret\_val;**

**}**

**/\* yy\_get\_previous\_state - get the state just before the EOB char was reached \*/**

**static yy\_state\_type yy\_get\_previous\_state()**

**{**

**register yy\_state\_type yy\_current\_state;**

**register char \*yy\_cp;**

**yy\_current\_state = yy\_start;**

**for ( yy\_cp = yytext\_ptr + YY\_MORE\_ADJ; yy\_cp < yy\_c\_buf\_p; ++yy\_cp )**

**{**

**register YY\_CHAR yy\_c = (\*yy\_cp ? yy\_ec[YY\_SC\_TO\_UI(\*yy\_cp)] : 1);**

**if ( yy\_accept[yy\_current\_state] )**

**{**

**yy\_last\_accepting\_state = yy\_current\_state;**

**yy\_last\_accepting\_cpos = yy\_cp;**

**}**

**while ( yy\_chk[yy\_base[yy\_current\_state] + yy\_c] != yy\_current\_state )**

**{**

**yy\_current\_state = (int) yy\_def[yy\_current\_state];**

**if ( yy\_current\_state >= 140 )**

**yy\_c = yy\_meta[(unsigned int) yy\_c];**

**}**

**yy\_current\_state = yy\_nxt[yy\_base[yy\_current\_state] + (unsigned int) yy\_c];**

**}**

**return yy\_current\_state;**

**}**

**/\* yy\_try\_NUL\_trans - try to make a transition on the NUL character**

**\***

**\* synopsis**

**\* next\_state = yy\_try\_NUL\_trans( current\_state );**

**\*/**

**#ifdef YY\_USE\_PROTOS**

**static yy\_state\_type yy\_try\_NUL\_trans( yy\_state\_type yy\_current\_state )**

**#else**

**static yy\_state\_type yy\_try\_NUL\_trans( yy\_current\_state )**

**yy\_state\_type yy\_current\_state;**

**#endif**

**{**

**register int yy\_is\_jam;**

**register char \*yy\_cp = yy\_c\_buf\_p;**

**register YY\_CHAR yy\_c = 1;**

**if ( yy\_accept[yy\_current\_state] )**

**{**

**yy\_last\_accepting\_state = yy\_current\_state;**

**yy\_last\_accepting\_cpos = yy\_cp;**

**}**

**while ( yy\_chk[yy\_base[yy\_current\_state] + yy\_c] != yy\_current\_state )**

**{**

**yy\_current\_state = (int) yy\_def[yy\_current\_state];**

**if ( yy\_current\_state >= 140 )**

**yy\_c = yy\_meta[(unsigned int) yy\_c];**

**}**

**yy\_current\_state = yy\_nxt[yy\_base[yy\_current\_state] + (unsigned int) yy\_c];**

**yy\_is\_jam = (yy\_current\_state == 139);**

**return yy\_is\_jam ? 0 : yy\_current\_state;**

**}**

**#ifndef YY\_NO\_UNPUT**

**#ifdef YY\_USE\_PROTOS**

**static void yyunput( int c, register char \*yy\_bp )**

**#else**

**static void yyunput( c, yy\_bp )**

**int c;**

**register char \*yy\_bp;**

**#endif**

**{**

**register char \*yy\_cp = yy\_c\_buf\_p;**

**/\* undo effects of setting up yytext \*/**

**\*yy\_cp = yy\_hold\_char;**

**if ( yy\_cp < yy\_current\_buffer->yy\_ch\_buf + 2 )**

**{ /\* need to shift things up to make room \*/**

**/\* +2 for EOB chars. \*/**

**register int number\_to\_move = yy\_n\_chars + 2;**

**register char \*dest = &yy\_current\_buffer->yy\_ch\_buf[**

**yy\_current\_buffer->yy\_buf\_size + 2];**

**register char \*source =**

**&yy\_current\_buffer->yy\_ch\_buf[number\_to\_move];**

**while ( source > yy\_current\_buffer->yy\_ch\_buf )**

**\*--dest = \*--source;**

**yy\_cp += (int) (dest - source);**

**yy\_bp += (int) (dest - source);**

**yy\_current\_buffer->yy\_n\_chars =**

**yy\_n\_chars = yy\_current\_buffer->yy\_buf\_size;**

**if ( yy\_cp < yy\_current\_buffer->yy\_ch\_buf + 2 )**

**YY\_FATAL\_ERROR( "flex scanner push-back overflow" );**

**}**

**\*--yy\_cp = (char) c;**

**yytext\_ptr = yy\_bp;**

**yy\_hold\_char = \*yy\_cp;**

**yy\_c\_buf\_p = yy\_cp;**

**}**

**#endif /\* ifndef YY\_NO\_UNPUT \*/**

**#ifdef \_\_cplusplus**

**static int yyinput()**

**#else**

**static int input()**

**#endif**

**{**

**int c;**

**\*yy\_c\_buf\_p = yy\_hold\_char;**

**if ( \*yy\_c\_buf\_p == YY\_END\_OF\_BUFFER\_CHAR )**

**{**

**/\* yy\_c\_buf\_p now points to the character we want to return.**

**\* If this occurs \*before\* the EOB characters, then it's a**

**\* valid NUL; if not, then we've hit the end of the buffer.**

**\*/**

**if ( yy\_c\_buf\_p < &yy\_current\_buffer->yy\_ch\_buf[yy\_n\_chars] )**

**/\* This was really a NUL. \*/**

**\*yy\_c\_buf\_p = '\0';**

**else**

**{ /\* need more input \*/**

**int offset = yy\_c\_buf\_p - yytext\_ptr;**

**++yy\_c\_buf\_p;**

**switch ( yy\_get\_next\_buffer() )**

**{**

**case EOB\_ACT\_LAST\_MATCH:**

**/\* This happens because yy\_g\_n\_b()**

**\* sees that we've accumulated a**

**\* token and flags that we need to**

**\* try matching the token before**

**\* proceeding. But for input(),**

**\* there's no matching to consider.**

**\* So convert the EOB\_ACT\_LAST\_MATCH**

**\* to EOB\_ACT\_END\_OF\_FILE.**

**\*/**

**/\* Reset buffer status. \*/**

**yyrestart( yyin );**

**/\* fall through \*/**

**case EOB\_ACT\_END\_OF\_FILE:**

**{**

**if ( yywrap() )**

**return EOF;**

**if ( ! yy\_did\_buffer\_switch\_on\_eof )**

**YY\_NEW\_FILE;**

**#ifdef \_\_cplusplus**

**return yyinput();**

**#else**

**return input();**

**#endif**

**}**

**case EOB\_ACT\_CONTINUE\_SCAN:**

**yy\_c\_buf\_p = yytext\_ptr + offset;**

**break;**

**}**

**}**

**}**

**c = \*(unsigned char \*) yy\_c\_buf\_p; /\* cast for 8-bit char's \*/**

**\*yy\_c\_buf\_p = '\0'; /\* preserve yytext \*/**

**yy\_hold\_char = \*++yy\_c\_buf\_p;**

**return c;**

**}**

**#ifdef YY\_USE\_PROTOS**

**void yyrestart( FILE \*input\_file )**

**#else**

**void yyrestart( input\_file )**

**FILE \*input\_file;**

**#endif**

**{**

**if ( ! yy\_current\_buffer )**

**yy\_current\_buffer = yy\_create\_buffer( yyin, YY\_BUF\_SIZE );**

**yy\_init\_buffer( yy\_current\_buffer, input\_file );**

**yy\_load\_buffer\_state();**

**}**

**#ifdef YY\_USE\_PROTOS**

**void yy\_switch\_to\_buffer( YY\_BUFFER\_STATE new\_buffer )**

**#else**

**void yy\_switch\_to\_buffer( new\_buffer )**

**YY\_BUFFER\_STATE new\_buffer;**

**#endif**

**{**

**if ( yy\_current\_buffer == new\_buffer )**

**return;**

**if ( yy\_current\_buffer )**

**{**

**/\* Flush out information for old buffer. \*/**

**\*yy\_c\_buf\_p = yy\_hold\_char;**

**yy\_current\_buffer->yy\_buf\_pos = yy\_c\_buf\_p;**

**yy\_current\_buffer->yy\_n\_chars = yy\_n\_chars;**

**}**

**yy\_current\_buffer = new\_buffer;**

**yy\_load\_buffer\_state();**

**/\* We don't actually know whether we did this switch during**

**\* EOF (yywrap()) processing, but the only time this flag**

**\* is looked at is after yywrap() is called, so it's safe**

**\* to go ahead and always set it.**

**\*/**

**yy\_did\_buffer\_switch\_on\_eof = 1;**

**}**

**#ifdef YY\_USE\_PROTOS**

**void yy\_load\_buffer\_state( void )**

**#else**

**void yy\_load\_buffer\_state()**

**#endif**

**{**

**yy\_n\_chars = yy\_current\_buffer->yy\_n\_chars;**

**yytext\_ptr = yy\_c\_buf\_p = yy\_current\_buffer->yy\_buf\_pos;**

**yyin = yy\_current\_buffer->yy\_input\_file;**

**yy\_hold\_char = \*yy\_c\_buf\_p;**

**}**

**#ifdef YY\_USE\_PROTOS**

**YY\_BUFFER\_STATE yy\_create\_buffer( FILE \*file, int size )**

**#else**

**YY\_BUFFER\_STATE yy\_create\_buffer( file, size )**

**FILE \*file;**

**int size;**

**#endif**

**{**

**YY\_BUFFER\_STATE b;**

**b = (YY\_BUFFER\_STATE) yy\_flex\_alloc( sizeof( struct yy\_buffer\_state ) );**

**if ( ! b )**

**YY\_FATAL\_ERROR( "out of dynamic memory in yy\_create\_buffer()" );**

**b->yy\_buf\_size = size;**

**/\* yy\_ch\_buf has to be 2 characters longer than the size given because**

**\* we need to put in 2 end-of-buffer characters.**

**\*/**

**b->yy\_ch\_buf = (char \*) yy\_flex\_alloc( b->yy\_buf\_size + 2 );**

**if ( ! b->yy\_ch\_buf )**

**YY\_FATAL\_ERROR( "out of dynamic memory in yy\_create\_buffer()" );**

**b->yy\_is\_our\_buffer = 1;**

**yy\_init\_buffer( b, file );**

**return b;**

**}**

**#ifdef YY\_USE\_PROTOS**

**void yy\_delete\_buffer( YY\_BUFFER\_STATE b )**

**#else**

**void yy\_delete\_buffer( b )**

**YY\_BUFFER\_STATE b;**

**#endif**

**{**

**if ( ! b )**

**return;**

**if ( b == yy\_current\_buffer )**

**yy\_current\_buffer = (YY\_BUFFER\_STATE) 0;**

**if ( b->yy\_is\_our\_buffer )**

**yy\_flex\_free( (void \*) b->yy\_ch\_buf );**

**yy\_flex\_free( (void \*) b );**

**}**

**#ifndef YY\_ALWAYS\_INTERACTIVE**

**#ifndef YY\_NEVER\_INTERACTIVE**

**extern int isatty YY\_PROTO(( int ));**

**#endif**

**#endif**

**#ifdef YY\_USE\_PROTOS**

**void yy\_init\_buffer( YY\_BUFFER\_STATE b, FILE \*file )**

**#else**

**void yy\_init\_buffer( b, file )**

**YY\_BUFFER\_STATE b;**

**FILE \*file;**

**#endif**

**{**

**yy\_flush\_buffer( b );**

**b->yy\_input\_file = file;**

**b->yy\_fill\_buffer = 1;**

**#if YY\_ALWAYS\_INTERACTIVE**

**b->yy\_is\_interactive = 1;**

**#else**

**#if YY\_NEVER\_INTERACTIVE**

**b->yy\_is\_interactive = 0;**

**#else**

**b->yy\_is\_interactive = file ? (isatty( fileno(file) ) > 0) : 0;**

**#endif**

**#endif**

**}**

**#ifdef YY\_USE\_PROTOS**

**void yy\_flush\_buffer( YY\_BUFFER\_STATE b )**

**#else**

**void yy\_flush\_buffer( b )**

**YY\_BUFFER\_STATE b;**

**#endif**

**{**

**if ( ! b )**

**return;**

**b->yy\_n\_chars = 0;**

**/\* We always need two end-of-buffer characters. The first causes**

**\* a transition to the end-of-buffer state. The second causes**

**\* a jam in that state.**

**\*/**

**b->yy\_ch\_buf[0] = YY\_END\_OF\_BUFFER\_CHAR;**

**b->yy\_ch\_buf[1] = YY\_END\_OF\_BUFFER\_CHAR;**

**b->yy\_buf\_pos = &b->yy\_ch\_buf[0];**

**b->yy\_at\_bol = 1;**

**b->yy\_buffer\_status = YY\_BUFFER\_NEW;**

**if ( b == yy\_current\_buffer )**

**yy\_load\_buffer\_state();**

**}**

**#ifndef YY\_NO\_SCAN\_BUFFER**

**#ifdef YY\_USE\_PROTOS**

**YY\_BUFFER\_STATE yy\_scan\_buffer( char \*base, yy\_size\_t size )**

**#else**

**YY\_BUFFER\_STATE yy\_scan\_buffer( base, size )**

**char \*base;**

**yy\_size\_t size;**

**#endif**

**{**

**YY\_BUFFER\_STATE b;**

**if ( size < 2 ||**

**base[size-2] != YY\_END\_OF\_BUFFER\_CHAR ||**

**base[size-1] != YY\_END\_OF\_BUFFER\_CHAR )**

**/\* They forgot to leave room for the EOB's. \*/**

**return 0;**

**b = (YY\_BUFFER\_STATE) yy\_flex\_alloc( sizeof( struct yy\_buffer\_state ) );**

**if ( ! b )**

**YY\_FATAL\_ERROR( "out of dynamic memory in yy\_scan\_buffer()" );**

**b->yy\_buf\_size = size - 2; /\* "- 2" to take care of EOB's \*/**

**b->yy\_buf\_pos = b->yy\_ch\_buf = base;**

**b->yy\_is\_our\_buffer = 0;**

**b->yy\_input\_file = 0;**

**b->yy\_n\_chars = b->yy\_buf\_size;**

**b->yy\_is\_interactive = 0;**

**b->yy\_at\_bol = 1;**

**b->yy\_fill\_buffer = 0;**

**b->yy\_buffer\_status = YY\_BUFFER\_NEW;**

**yy\_switch\_to\_buffer( b );**

**return b;**

**}**

**#endif**

**#ifndef YY\_NO\_SCAN\_STRING**

**#ifdef YY\_USE\_PROTOS**

**YY\_BUFFER\_STATE yy\_scan\_string( yyconst char \*yy\_str )**

**#else**

**YY\_BUFFER\_STATE yy\_scan\_string( yy\_str )**

**yyconst char \*yy\_str;**

**#endif**

**{**

**int len;**

**for ( len = 0; yy\_str[len]; ++len )**

**;**

**return yy\_scan\_bytes( yy\_str, len );**

**}**

**#endif**

**#ifndef YY\_NO\_SCAN\_BYTES**

**#ifdef YY\_USE\_PROTOS**

**YY\_BUFFER\_STATE yy\_scan\_bytes( yyconst char \*bytes, int len )**

**#else**

**YY\_BUFFER\_STATE yy\_scan\_bytes( bytes, len )**

**yyconst char \*bytes;**

**int len;**

**#endif**

**{**

**YY\_BUFFER\_STATE b;**

**char \*buf;**

**yy\_size\_t n;**

**int i;**

**/\* Get memory for full buffer, including space for trailing EOB's. \*/**

**n = len + 2;**

**buf = (char \*) yy\_flex\_alloc( n );**

**if ( ! buf )**

**YY\_FATAL\_ERROR( "out of dynamic memory in yy\_scan\_bytes()" );**

**for ( i = 0; i < len; ++i )**

**buf[i] = bytes[i];**

**buf[len] = buf[len+1] = YY\_END\_OF\_BUFFER\_CHAR;**

**b = yy\_scan\_buffer( buf, n );**

**if ( ! b )**

**YY\_FATAL\_ERROR( "bad buffer in yy\_scan\_bytes()" );**

**/\* It's okay to grow etc. this buffer, and we should throw it**

**\* away when we're done.**

**\*/**

**b->yy\_is\_our\_buffer = 1;**

**return b;**

**}**

**#endif**

**#ifndef YY\_NO\_PUSH\_STATE**

**#ifdef YY\_USE\_PROTOS**

**static void yy\_push\_state( int new\_state )**

**#else**

**static void yy\_push\_state( new\_state )**

**int new\_state;**

**#endif**

**{**

**if ( yy\_start\_stack\_ptr >= yy\_start\_stack\_depth )**

**{**

**yy\_size\_t new\_size;**

**yy\_start\_stack\_depth += YY\_START\_STACK\_INCR;**

**new\_size = yy\_start\_stack\_depth \* sizeof( int );**

**if ( ! yy\_start\_stack )**

**yy\_start\_stack = (int \*) yy\_flex\_alloc( new\_size );**

**else**

**yy\_start\_stack = (int \*) yy\_flex\_realloc(**

**(void \*) yy\_start\_stack, new\_size );**

**if ( ! yy\_start\_stack )**

**YY\_FATAL\_ERROR(**

**"out of memory expanding start-condition stack" );**

**}**

**yy\_start\_stack[yy\_start\_stack\_ptr++] = YY\_START;**

**BEGIN(new\_state);**

**}**

**#endif**

**#ifndef YY\_NO\_POP\_STATE**

**static void yy\_pop\_state()**

**{**

**if ( --yy\_start\_stack\_ptr < 0 )**

**YY\_FATAL\_ERROR( "start-condition stack underflow" );**

**BEGIN(yy\_start\_stack[yy\_start\_stack\_ptr]);**

**}**

**#endif**

**#ifndef YY\_NO\_TOP\_STATE**

**static int yy\_top\_state()**

**{**

**return yy\_start\_stack[yy\_start\_stack\_ptr - 1];**

**}**

**#endif**

**#ifndef YY\_EXIT\_FAILURE**

**#define YY\_EXIT\_FAILURE 2**

**#endif**

**#ifdef YY\_USE\_PROTOS**

**static void yy\_fatal\_error( yyconst char msg[] )**

**#else**

**static void yy\_fatal\_error( msg )**

**char msg[];**

**#endif**

**{**

**(void) fprintf( stderr, "%s\n", msg );**

**exit( YY\_EXIT\_FAILURE );**

**}**

**/\* Redefine yyless() so it works in section 3 code. \*/**

**#undef yyless**

**#define yyless(n) \**

**do \**

**{ \**

**/\* Undo effects of setting up yytext. \*/ \**

**yytext[yyleng] = yy\_hold\_char; \**

**yy\_c\_buf\_p = yytext + n; \**

**yy\_hold\_char = \*yy\_c\_buf\_p; \**

**\*yy\_c\_buf\_p = '\0'; \**

**yyleng = n; \**

**} \**

**while ( 0 )**

**/\* Internal utility routines. \*/**

**#ifndef yytext\_ptr**

**#ifdef YY\_USE\_PROTOS**

**static void yy\_flex\_strncpy( char \*s1, yyconst char \*s2, int n )**

**#else**

**static void yy\_flex\_strncpy( s1, s2, n )**

**char \*s1;**

**yyconst char \*s2;**

**int n;**

**#endif**

**{**

**register int i;**

**for ( i = 0; i < n; ++i )**

**s1[i] = s2[i];**

**}**

**#endif**

**#ifdef YY\_NEED\_STRLEN**

**#ifdef YY\_USE\_PROTOS**

**static int yy\_flex\_strlen( yyconst char \*s )**

**#else**

**static int yy\_flex\_strlen( s )**

**yyconst char \*s;**

**#endif**

**{**

**register int n;**

**for ( n = 0; s[n]; ++n )**

**;**

**return n;**

**}**

**#endif**

**#ifdef YY\_USE\_PROTOS**

**static void \*yy\_flex\_alloc( yy\_size\_t size )**

**#else**

**static void \*yy\_flex\_alloc( size )**

**yy\_size\_t size;**

**#endif**

**{**

**return (void \*) malloc( size );**

**}**

**#ifdef YY\_USE\_PROTOS**

**static void \*yy\_flex\_realloc( void \*ptr, yy\_size\_t size )**

**#else**

**static void \*yy\_flex\_realloc( ptr, size )**

**void \*ptr;**

**yy\_size\_t size;**

**#endif**

**{**

**/\* The cast to (char \*) in the following accommodates both**

**\* implementations that use char\* generic pointers, and those**

**\* that use void\* generic pointers. It works with the latter**

**\* because both ANSI C and C++ allow castless assignment from**

**\* any pointer type to void\*, and deal with argument conversions**

**\* as though doing an assignment.**

**\*/**

**return (void \*) realloc( (char \*) ptr, size );**

**}**

**#ifdef YY\_USE\_PROTOS**

**static void yy\_flex\_free( void \*ptr )**

**#else**

**static void yy\_flex\_free( ptr )**

**void \*ptr;**

**#endif**

**{**

**free( ptr );**

**}**

**#if YY\_MAIN**

**int main()**

**{**

**yylex();**

**return 0;**

**}**

**#endif**

**#line 45 "zf.l"**

**Upper(char \*s,int l)**

**{ int i;**

**ccount=ccount+l;**

**wcount=wcount+1;**

**for(i=0;i<l;i++){s[i]=toupper(s[i]); }**

**}**

**void comment()**

**{**

**char ch=input();**

**while (ch!='\*')**

**{printf("%c",ch);**

**ch=input();**

**}**

**while(ch!='/')**

**{printf("%c",ch);**

**ch=input();**

**}**

**printf("/");**

**return;**

**}**

**main(void) {**

**FILE \*infile;**

**char infileName[200];**

**printf("Please input the file Name: ");**

**scanf("%s",&infileName);**

**infile=fopen(infileName,"r");**

**yyin=infile;**

**printf("开始词法分析： \n");**

**yylex();**

**lineno=lineno-1;**

**printf("Line:%d\n",lineno);**

**printf("word count:%d\n",wcount);**

**printf("char count:%d\n",ccount);**

**system(“pause”);**

**return 258;**

**}**