

Lang Processing - Program1

1.0.0

構築: Doxygen 1.8.20

1 データ構造索引	1
1.1 データ構造	1
2 ファイル索引	3
2.1 ファイル一覧	3
3 データ構造詳解	5
3.1 ID 構造体	5
3.1.1 詳解	5
3.1.2 フィールド詳解	5
3.1.2.1 count	5
3.1.2.2 name	6
3.1.2.3 nextp	6
3.2 KEY 構造体	6
3.2.1 詳解	6
3.2.2 フィールド詳解	6
3.2.2.1 keytoken	6
3.2.2.2 keyword	6
4 ファイル詳解	7
4.1 id-list.c ファイル	7
4.1.1 関数詳解	8
4.1.1.1 id_countup()	8
4.1.1.2 init_idtab()	8
4.1.1.3 print_idtab()	9
4.1.1.4 release_idtab()	9
4.1.1.5 search_idtab()	10
4.1.2 変数詳解	10
4.1.2.1 idroot	10
4.2 scan.c ファイル	10
4.2.1 関数詳解	12
4.2.1.1 _isblank()	12
4.2.1.2 end_scan()	13
4.2.1.3 get_keyword_token_code()	13
4.2.1.4 get_linenum()	14
4.2.1.5 init_scan()	14
4.2.1.6 look_ahead()	15
4.2.1.7 scan()	16
4.2.1.8 scan_alnum()	18
4.2.1.9 scan_comment()	19
4.2.1.10 scan_digit()	20
4.2.1.11 scan_string()	21
4.2.1.12 scan_symbol()	22

4.2.1.13 set_token_linenum()	23
4.2.1.14 string_attr_push_back()	23
4.2.2 変数詳解	24
4.2.2.1 current_char	24
4.2.2.2 fp	25
4.2.2.3 linenum	25
4.2.2.4 next_char	25
4.2.2.5 num_attr	25
4.2.2.6 string_attr	25
4.2.2.7 token_linenum	26
4.3 token-list.c ファイル	26
4.3.1 関数詳解	26
4.3.1.1 error()	27
4.3.1.2 main()	28
4.3.2 変数詳解	29
4.3.2.1 key	29
4.3.2.2 numtoken	30
4.3.2.3 tokenstr	30
4.4 token-list.h ファイル	31
4.4.1 マクロ定義詳解	33
4.4.1.1 KEYWORDSIZE	33
4.4.1.2 MAX_NUM_ATTR	33
4.4.1.3 MAXSTRSIZE	33
4.4.1.4 NUMOFTOKEN	34
4.4.1.5 TAND	34
4.4.1.6 TARRAY	34
4.4.1.7 TASSIGN	34
4.4.1.8 TBEGIN	34
4.4.1.9 TBOOLEAN	35
4.4.1.10 TBREAK	35
4.4.1.11 TCALL	35
4.4.1.12 TCHAR	35
4.4.1.13 TCOLON	35
4.4.1.14 TCOMMA	36
4.4.1.15 TDIV	36
4.4.1.16 TDO	36
4.4.1.17 TDOT	36
4.4.1.18 TELSE	36
4.4.1.19 TEND	37
4.4.1.20 TEQUAL	37
4.4.1.21 TFALSE	37
4.4.1.22 TGR	37

4.4.1.23 TGREQ	37
4.4.1.24 TIF	38
4.4.1.25 TINTEGER	38
4.4.1.26 TLE	38
4.4.1.27 TLEEQ	38
4.4.1.28 TLPAREN	38
4.4.1.29 TLSQPAREN	39
4.4.1.30 TMINUS	39
4.4.1.31 TNAME	39
4.4.1.32 TNOT	39
4.4.1.33 TNOTEQ	39
4.4.1.34 TNUMBER	40
4.4.1.35 TOF	40
4.4.1.36 TOR	40
4.4.1.37 TPLUS	40
4.4.1.38 TPROCEDURE	40
4.4.1.39 TPROGRAM	41
4.4.1.40 TREAD	41
4.4.1.41 TREADLN	41
4.4.1.42 TRETURN	41
4.4.1.43 TRPAREN	41
4.4.1.44 TRSQPAREN	42
4.4.1.45 TSEMI	42
4.4.1.46 TSTAR	42
4.4.1.47 TSTRING	42
4.4.1.48 TTHEN	42
4.4.1.49 TTRUE	43
4.4.1.50 TVAR	43
4.4.1.51 TWHILE	43
4.4.1.52 TWRITE	43
4.4.1.53 TWRITELN	43
4.4.2 関数詳解	44
4.4.2.1 end_scan()	44
4.4.2.2 error()	44
4.4.2.3 get_linenum()	45
4.4.2.4 init_scan()	45
4.4.2.5 scan()	46
4.4.3 変数詳解	48
4.4.3.1 fp	48
4.4.3.2 key	48
4.4.3.3 num_attr	48
4.4.3.4 string_attr	48

Chapter 1

データ構造索引

1.1 データ構造

データ構造一覧です。

ID	5
KEY	6
A pair of token codes for a keyword	6

Chapter 2

ファイル索引

2.1 ファイル一覧

ファイル一覧です。

id-list.c	7
scan.c	10
token-list.c	26
token-list.h	31

Chapter 3

データ構造詳解

3.1 ID 構造体

ID 連携図



フィールド

- char * name
- int count
- struct ID * nextp

3.1.1 詳解

id-list.c の 3 行目に定義があります。

3.1.2 フィールド詳解

3.1.2.1 count

```
int ID::count
```

id-list.c の 5 行目に定義があります。

3.1.2.2 name

```
char* ID::name
```

id-list.c の 4 行目に定義があります。

3.1.2.3 nextp

```
struct ID* ID::nextp
```

id-list.c の 6 行目に定義があります。

この構造体詳解は次のファイルから抽出されました:

- [id-list.c](#)

3.2 KEY 構造体

A pair of token codes for a keyword

```
#include <token-list.h>
```

フィールド

- char * [keyword](#)
- int [keytoken](#)

3.2.1 詳解

A pair of token codes for a keyword

token-list.h の 127 行目に定義があります。

3.2.2 フィールド詳解

3.2.2.1 keytoken

```
int KEY::keytoken
```

keyword strings

token-list.h の 129 行目に定義があります。

3.2.2.2 keyword

```
char* KEY::keyword
```

token-list.h の 128 行目に定義があります。

この構造体詳解は次のファイルから抽出されました:

- [token-list.h](#)

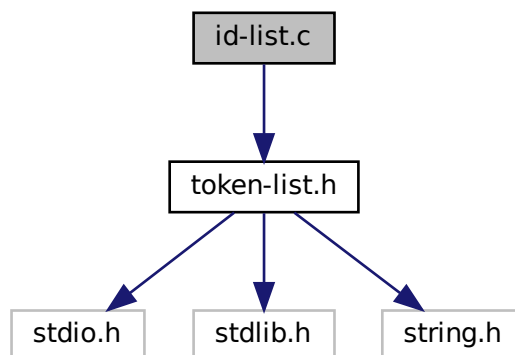
Chapter 4

ファイル詳解

4.1 id-list.c ファイル

```
#include "token-list.h"
```

id-list.c の依存先関係図:



データ構造

- struct `ID`

関数

- void `init_idtab ()`
- struct `ID * search_idtab (char *np)`
- void `id_countup (char *np)`
- void `print_idtab ()`
- void `release_idtab ()`

変数

- struct ID * idroot

4.1.1 関数詳解

4.1.1.1 id_countup()

```
void id_countup (
    char * np )
```

id-list.c の 22 行目に定義があります。

```
22                                     { /* Register and count up the name pointed by np */
23     struct ID *p;
24     char *cp;
25
26     if ((p = search_idtab(np)) != NULL)
27         p->count++;
28     else {
29         if ((p = (struct ID *)malloc(sizeof(struct ID))) == NULL) {
30             printf("can not malloc in id_countup\n");
31             return;
32         }
33         if ((cp = (char *)malloc(strlen(np) + 1)) == NULL) {
34             printf("can not malloc-2 in id_countup\n");
35             return;
36         }
37         strcpy(cp, np);
38         p->name = cp;
39         p->count = 1;
40         p->nextp = idroot;
41         idroot = p;
42     }
43 }
```

呼び出し関係図:



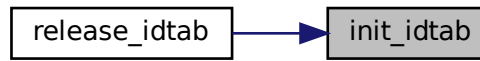
4.1.1.2 init_idtab()

```
void init_idtab ( )
```

id-list.c の 9 行目に定義があります。

```
9     { /* Initialise the table */
10     idroot = NULL;
11 }
```

被呼び出し関係図:



4.1.1.3 print_idtab()

```
void print_idtab ( )
```

id-list.c の 45 行目に定義があります。

```

45     { /* Output the registered data */
46     struct ID *p;
47
48     for (p = idroot; p != NULL; p = p->nextp) {
49         if (p->count != 0)
50             printf("\t\"Identifier\" \"%s\" \t%d\n", p->name, p->count);
51     }
52 }
```

4.1.1.4 release_idtab()

```
void release_idtab ( )
```

id-list.c の 54 行目に定義があります。

```

54     { /* Release the data structure */
55     struct ID *p, *q;
56
57     for (p = idroot; p != NULL; p = q) {
58         free(p->name);
59         q = p->nextp;
60         free(p);
61     }
62     init_idtab();
63 }
```

呼び出し関係図:



4.1.1.5 search_idtab()

```
struct ID* search_idtab (  
    char * np )
```

id-list.c の 13 行目に定義があります。

```
13      { /* search the name pointed by np */  
14      struct ID *p;  
15  
16      for (p = idroot; p != NULL; p = p->nextp) {  
17          if (strcmp(np, p->name) == 0) return (p);  
18      }  
19      return (NULL);  
20 }
```

被呼び出し関係図:



4.1.2 変数詳解

4.1.2.1 idroot

```
struct ID * idroot
```

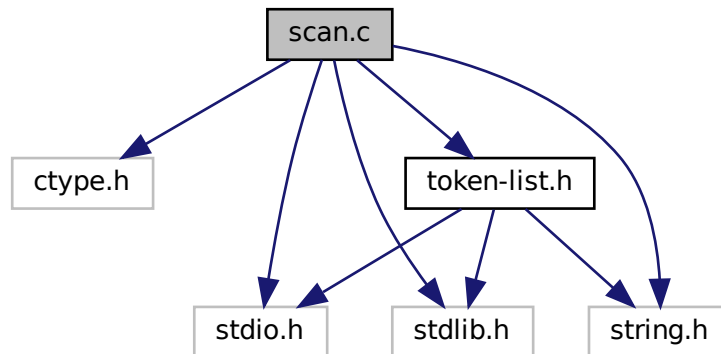
4.2 scan.c ファイル

```
#include <ctype.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>
```



```
#include "token-list.h"
```

scan.c の依存先関係図:



変数

- FILE * `fp`
- int `num_attr` = 0
- char `string_attr` [MAXSTRSIZE]

looka ahead

- static int `current_char`
- static int `next_char` = '\0'
- static int `linenum` = 1
- static int `token_linenum` = 0
- static void `look_ahead` ()
Pre-reading file
- static int `_isblank` (int c)
Determine if a character is a space character or not.
- int `get_linenum` (void)
Return the line number of the last token scanned
- void `set_token_linenum` (void)
Set the line number of the last token scanned
- static int `scan_alnum` ()
Scan one string of letters and numbers
- static int `scan_digit` ()
Scan one number sequence.
- static int `scan_string` ()
Scan one string.
- static int `scan_comment` ()
Scan the annotation
- static int `scan_symbol` ()
Scan one symbol

- static int `get.keyword.token.code` (char *token)
Get the token code for a token
- static int `string.attr.push.back` (const char c)
Adding characters to the end of a scanned string
- int `init.scan` (char *filename)
Initialization to begin scanning
- int `scan` (void)
Scan the file and return the token code
- int `end.scan` (void)
The process of finishing the scan

4.2.1 関数詳解

4.2.1.1 `_isblank()`

```
static int _isblank (
    int c ) [static]
```

Determine if a character is a space character or not.

引数

in	<i>c</i>	Character to be determined
----	----------	----------------------------

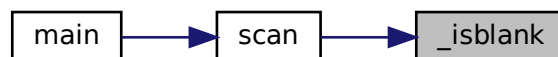
戻り値

int Returns 1 for a blank character, 0 otherwise.

`scan.c` の 135 行目に定義があります。

```
135     {
136     if (c == ' ' || c == '\t') {
137         return 1;
138     } else {
139         return 0;
140     }
141 }
```

被呼び出し関係図:



4.2.1.2 end_scan()

```
int end_scan (
    void )
```

The process of finishing the scan

戻り値

int Returns 0 on success and -1 on failure.

scan.c の 123 行目に定義があります。

```
123     {
124         if (fclose(fp) == EOF) {
125             return -1;
126         }
127         return 0;
128     }
```

被呼び出し関係図:



4.2.1.3 get_keyword_token_code()

```
static int get_keyword_token_code (
    char * token ) [static]
```

Get the token code for a token

引数

in	token	token to be determined
----	-------	------------------------

戻り値

int If it is a keyword, it returns its token code, otherwise it returns the Name token code

scan.c の 308 行目に定義があります。

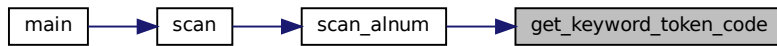
```
308     {
309         int index;
310         /* TODO: binary search */
311         for (index = 0; index < KEYWORDSIZE; index++) {
312             if (strcmp(token, key[index].keyword) == 0) {
```

```

313             /* This token is Keyword */
314             return key[index].keytoken;
315         }
316     }
317     /* This token is NAME*/
318     return TNAME;
319 }

```

被呼び出し関係図:



4.2.1.4 get_linenum()

```

int get_linenum (
    void )

```

Return the line number of the last token scanned

戻り値

int Return line number

scan.c の 108 行目に定義があります。

```

108     {
109         return token_linenum;
110     }

```

4.2.1.5 init_scan()

```

int init_scan (
    char * filename )

```

Initialization to begin scanning

引数

in	filename	File name to scan
----	----------	-------------------

戻り値

int Returns 0 on success and -1 on failure.

scan.c の 44 行目に定義があります。

```
44      {  
45      if ((fp = fopen(filename, "r")) == NULL) {  
46          return -1;  
47      }  
48        
49      look_ahead();  
50      look_ahead();  
51        
52      return 0;  
53 }
```

呼び出し関係図:



被呼び出し関係図:



4.2.1.6 look_ahead()

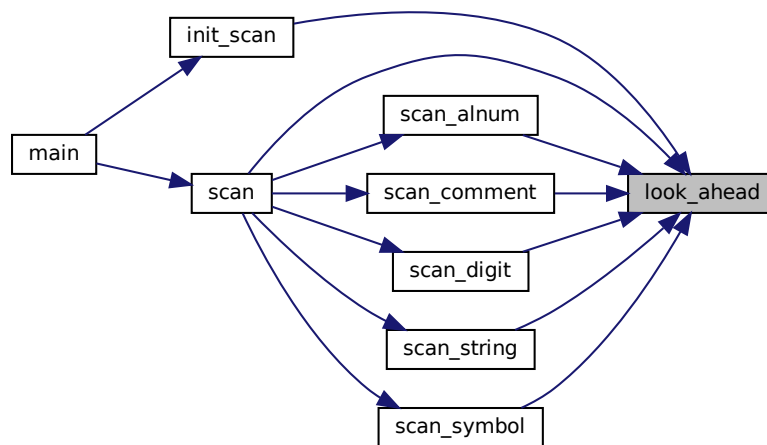
```
static void look_ahead ( ) [static]
```

Pre-reading file

scan.c の 341 行目に定義があります。

```
341      {  
342      current_char = next_char;  
343      next_char = fgetc(fp);  
344      return;  
345 }
```

被呼び出し関係図:



4.2.1.7 scan()

```
int scan (
    void )
```

Scan the file and return the token code

戻り値

int Returns token code on success and -1 on failure.

scan.c の 59 行目に定義があります。

```

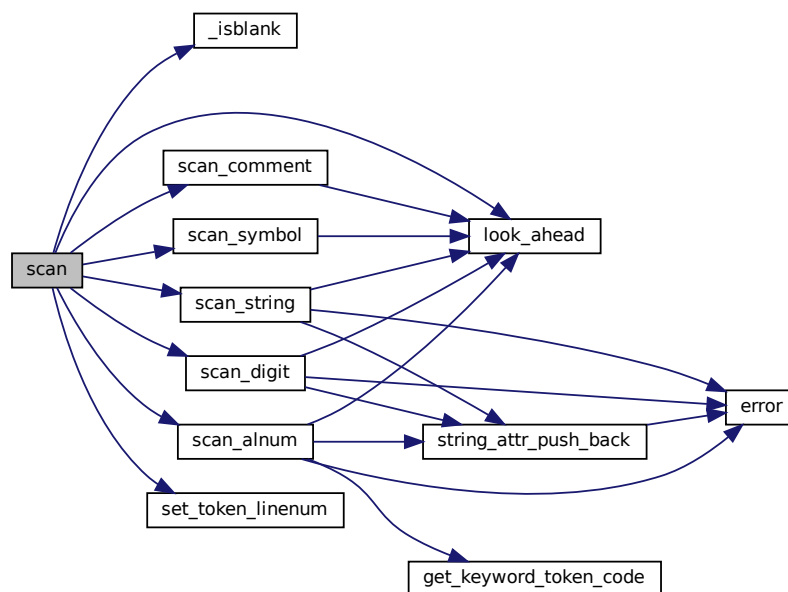
59 {
60     int token_code = -1;
61     while (1) {
62         if (current_char == EOF) { /* End Of File*/
63             return -1;
64         } else if (current_char == '\r' || current_char == '\n') { /* End of Line */
65             if (current_char == '\r') {
66                 if (next_char == '\n') {
67                     look_ahead();
68                 }
69                 look_ahead();
70                 linenum++;
71             } else {
72                 if (next_char == '\r') {
73                     look_ahead();
74                 }
75                 look_ahead();
76                 linenum++;
77             }
78         } else if (!isblank(current_char)) { /* Separator (Space or Tab) */
79             look_ahead();
80         } else if (!isprint(current_char)) { /* Not Graphic Character(0x20~0x7e) */
81             return -1;
82         } else if (isalpha(current_char)) { /* Name or Keyword */
83             token_code = scan_alnum();
84             break;
85         } else if (isdigit(current_char)) { /* Digit */
```

```

86         token_code = scan_digit();
87         break;
88     } else if (current_char == '\\') { /* String */
89         token_code = scan_string();
90         break;
91     } else if ((current_char == '/' && next_char == '*') || current_char == '{') { /* Comment */
92         if (scan_comment() == -1) {
93             break;
94         }
95     } else { /* Symbol */
96         token_code = scan_symbol();
97         break;
98     }
99 }
100 set_token_linenum();
101 return token_code;
102 }

```

呼び出し関係図:



被呼び出し関係図:



4.2.1.8 scan_alnum()

```
static int scan_alnum ( ) [static]
```

Scan one string of letters and numbers

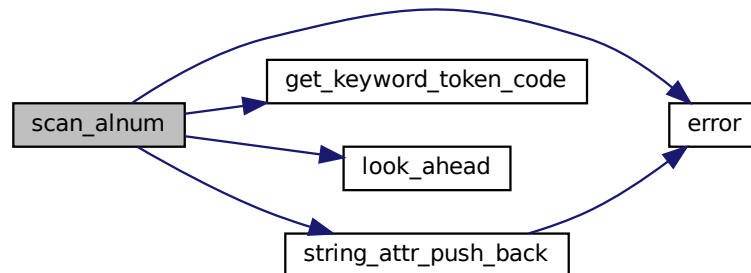
戻り値

int Returns token code on success and -1 on failure.

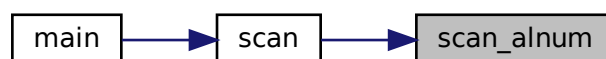
scan.c の 147 行目に定義があります。

```
147     {
148     memset(string_attr, '\0', sizeof(string_attr));
149     string_attr[0] = current_char;
150
151     look_ahead();
152     while (isalnum(current_char)) {
153         if (string_attr_push_back(current_char) == -1) {
154             error("function scan_alnum()");
155             return -1;
156         }
157         look_ahead();
158     }
159     return get_keyword_token_code(string_attr);
160 }
```

呼び出し関係図:



被呼び出し関係図:



4.2.1.9 scan_comment()

```
static int scan_comment ( ) [static]
```

Scan the annotation

戻り値

int Returns 0 on success and -1 on failure.

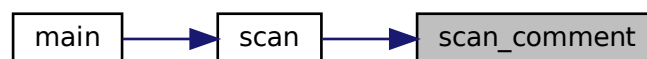
scan.c の 219 行目に定義があります。

```
219      {
220      if (current_char == '/' && next_char == '*') {
221          look_ahead();
222          look_ahead();
223          while (current_char != EOF) {
224              if (current_char == '*' && next_char == '/') {
225                  look_ahead();
226                  look_ahead();
227                  return 0;
228              }
229              look_ahead();
230          }
231      } else if (current_char == '{') {
232          look_ahead();
233          while (current_char != EOF) {
234              if (current_char == '}') {
235                  look_ahead();
236                  return 0;
237              }
238              look_ahead();
239          }
240      }
241      /* EOF */
242      return -1;
243  }
```

呼び出し関係図:



被呼び出し関係図:



4.2.1.10 scan_digit()

```
static int scan_digit ( ) [static]
```

Scan one number sequence.

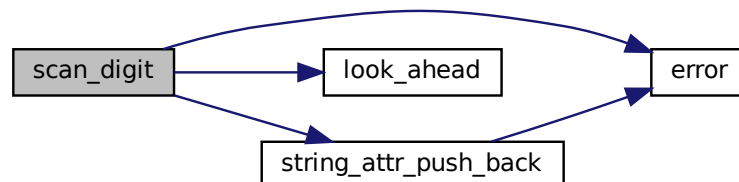
戻り値

int Returns token code of number on success and -1 on failure.

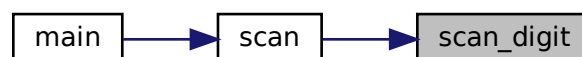
scan.c の 166 行目に定義があります。

```
166     {
167         int num = current_char - '0';
168
169         memset(string_attr, '\0', sizeof(string_attr));
170         string_attr[0] = current_char;
171
172         look_ahead();
173         while (isdigit(current_char)) {
174             if (string_attr_push_back(current_char) == -1) {
175                 error("function scan_digit()");
176                 return -1;
177             }
178             num *= 10;
179             num += current_char - '0';
180             look_ahead();
181         }
182         if (num <= MAX_NUM_ATTR) {
183             num_attr = num;
184             return TNUMBER;
185         }
186
187         return -1;
188     }
```

呼び出し関係図:



被呼び出し関係図:



4.2.1.11 scan_string()

```
static int scan_string ( ) [static]
```

Scan one string.

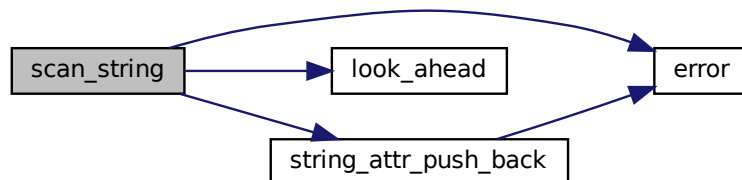
戻り値

int Returns token code of string on success and -1 on failure.

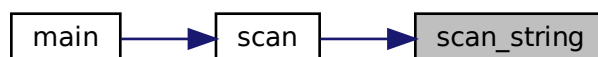
scan.c の 194 行目に定義があります。

```
194     {
195     memset(string_attr, '\0', sizeof(string_attr));
196     look_ahead();
197
198     while ((!(current_char == '\\' || next_char == '\\')) {
199         if (!isprint(current_char)) {
200             error("function scan_string()");
201             fprintf(stderr, "[%c]0x%x is not graphic character.\n", current_char, current_char);
202             return -1;
203         }
204         if (string_attr_push_back(current_char) == -1) {
205             error("function scan_digit()");
206             return -1;
207         }
208         look_ahead();
209     }
210     look_ahead(); /* read '\\' */
211
212     return TSTRING;
213 }
```

呼び出し関係図:



被呼び出し関係図:



4.2.1.12 scan_symbol()

```
static int scan_symbol ( ) [static]
```

Scan one symbol

戻り値

int Returns token code of symbol on success and -1 on failure.

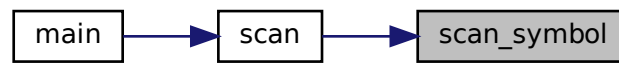
scan.c の 249 行目に定義があります。

```
249     {
250     char symbol = current_char;
251     look_ahead();
252     switch (symbol) {
253     case '+':
254         return TPLUS;
255     case '-':
256         return TMINUS;
257     case '*':
258         return TSTAR;
259     case '=':
260         return TEQUAL;
261     case '<':
262         if (current_char == '>') {
263             look_ahead();
264             return TNOTEQ;
265         } else if (current_char == '=') {
266             look_ahead();
267             return TLEEQ;
268         } else {
269             return TLE;
270         }
271     case '>':
272         if (current_char == '=') {
273             look_ahead();
274             return TGREQ;
275         } else {
276             return TGR;
277         }
278     case '(':
279         return TLPAREN;
280     case ')':
281         return TRPAREN;
282     case '[':
283         return TLSQPAREN;
284     case ']':
285         return TRSQPAREN;
286     case ':':
287         if (current_char == '=') {
288             look_ahead();
289             return TASSIGN;
290         } else {
291             return TCOLON;
292         }
293     case '.':
294         return TDOT;
295     case ',':
296         return TCOMMA;
297     case ';':
298         return TSEMI;
299     default:
300         return -1;
301     }
302 }
```

呼び出し関係図:



被呼び出し関係図:



4.2.1.13 set_token_linenum()

```
void set_token_linenum (  
    void )
```

Set the line number of the last token scanned

scan.c の 115 行目に定義があります。

```
115     {  
116         token_linenum = linenum;  
117     }
```

被呼び出し関係図:



4.2.1.14 string_attr_push_back()

```
static int string_attr_push_back (  
    const char c ) [static]
```

Adding characters to the end of a scanned string

引数

in	c	Characters to add
----	---	-------------------

戻り値

int Returns 0 on success and -1 on failure.

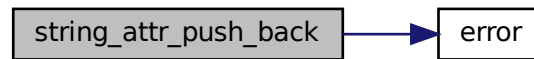
scan.c の 326 行目に定義があります。

```

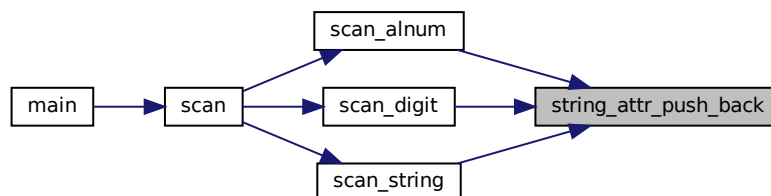
326                                     {
327     int len = strlen(string_attr);
328     if (len < MAXSTRSIZE - 1) {
329         string_attr[len] = c;
330         return 0;
331     } else {
332         /* Buffer Overflow */
333         error("string_attr: Buffer Overflow.");
334         return -1;
335     }
336 }

```

呼び出し関係図:



被呼び出し関係図:



4.2.2 変数詳解

4.2.2.1 current_char

```
int current_char [static]
```

The letters you just loaded.

scan.c の 18 行目に定義があります。

4.2.2.2 fp

```
FILE* fp
```

File pointer of the loaded file

scan.c の 9 行目に定義があります。

4.2.2.3 lineno

```
int lineno = 1 [static]
```

Line number of the character you just loaded

scan.c の 24 行目に定義があります。

4.2.2.4 next_char

```
int next_char = '\0' [static]
```

look-ahead character

scan.c の 20 行目に定義があります。

4.2.2.5 num_attr

```
int num_attr = 0
```

Scanned unsigned integer

scan.c の 11 行目に定義があります。

4.2.2.6 string_attr

```
char string_attr[MAXSTRSIZE]
```

Scanned string

scan.c の 13 行目に定義があります。

4.2.2.7 token.linenum

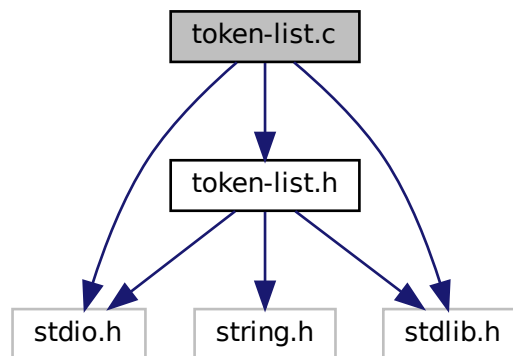
```
int token.linenum = 0 [static]
```

The line number of the last token scanned

scan.c の 26 行目に定義があります。

4.3 token-list.c ファイル

```
#include "token-list.h"  
#include <stdio.h>  
#include <stdlib.h>  
token-list.c の依存先関係図:
```



関数

- int `main` (int nc, char *np[])
main function
- void `error` (char *mes)
display an error message

変数

- struct `KEY` `key` [`KEYWORDSIZE`]
- int `numtoken` [`NUMOFTOKEN+1`]
- char * `tokenstr` [`NUMOFTOKEN+1`]

4.3.1 関数詳解

4.3.1.1 error()

```
void error (
    char * mes )
```

display an error message

引数

in	<i>mes</i>	Error message
----	------------	---------------

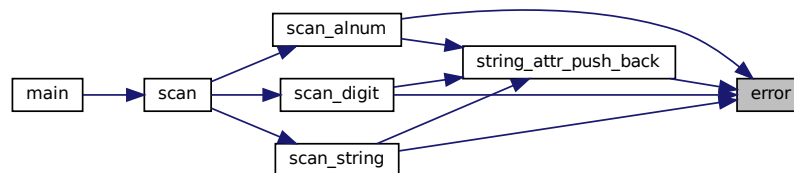
token-list.c の 94 行目に定義があります。

```

94     {
95     fprintf(stderr, "\n ERROR: %s\n", mes);
96 }

```

被呼び出し関係図:



4.3.1.2 main()

```

int main (
    int nc,
    char * np[] )

```

main function

引数

in	<i>nc</i>	The number of arguments
in	<i>np</i>	File name to read

戻り値

int Returns 0 on success and 1 on failure.

token-list.c の 57 行目に定義があります。

```

57     {
58     int token, index;
59
60     if (nc < 2) {
61         fprintf(stderr, "File name id not given.\n");
62         return EXIT_FAILURE;
63     }
64     if (init_scan(np[1]) < 0) {
65         fprintf(stderr, "File %s can not open.\n", np[1]);
66         return EXIT_FAILURE;
67     }
68
69     memset(numtoken, 0, sizeof(numtoken));
70

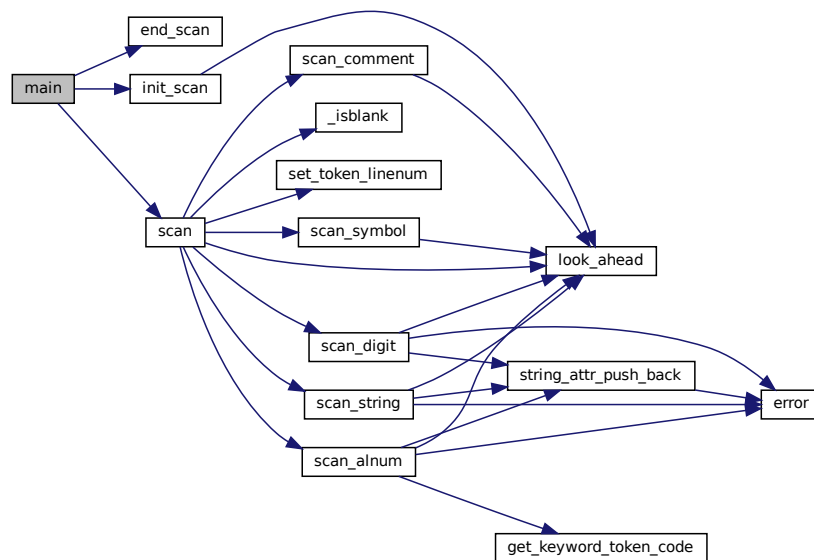
```

```

71     while ((token = scan()) >= 0) {
72         /* Count the tokens */
73         numtoken[token]++;
74     }
75
76     if (end_scan() < 0) {
77         fprintf(stderr, "File %s can not close.\n", np[1]);
78         return EXIT_FAILURE;
79     }
80     /* Output the results of the count. */
81     for (index = 0; index < NUMOFTOKEN + 1; index++) {
82         if (numtoken[index] > 0) {
83             fprintf(stdout, "%10s: %5d\n", tokenstr[index], numtoken[index]);
84         }
85     }
86
87     return 0;
88 }

```

呼び出し関係図:



4.3.2 変数詳解

4.3.2.1 key

```
struct KEY key[KEYWORDSIZE]
```

初期値:

```

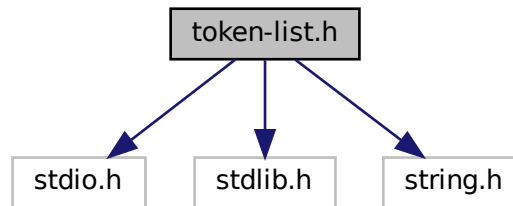
= {
    {"and", TAND},
    {"array", TARRAY},
    {"begin", TBEGIN},
    {"boolean", TBOOLEAN},
    {"break", TBREAK},
    {"call", TCALL},
    {"char", TCHAR},
    {"div", TDIV},
    {"do", TDO},
    {"else", TELSE},

```

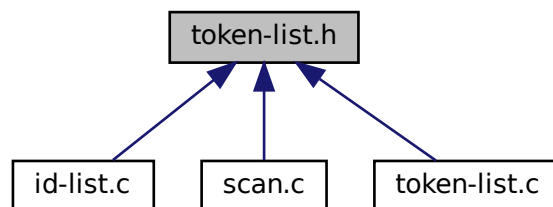

4.4 token-list.h ファイル

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

token-list.h の依存先関係図:



被依存関係図:



データ構造

- struct **KEY**

A pair of token codes for a keyword

マクロ定義

- #define **MAXSTRSIZE** 1024

definition of token code

- #define TNAME 1
- #define TPROGRAM 2
- #define TVAR 3
- #define TARRAY 4
- #define TOF 5
- #define TBEGIN 6
- #define TEND 7
- #define TIF 8
- #define TTHEN 9
- #define TELSE 10
- #define TPROCEDURE 11
- #define TRETURN 12
- #define TCALL 13
- #define TWHILE 14
- #define TDO 15
- #define TNOT 16
- #define TOR 17
- #define TDIV 18
- #define TAND 19
- #define TCHAR 20
- #define TINTEGER 21
- #define TBOOLEAN 22
- #define TREADLN 23
- #define TWRITELN 24
- #define TTRUE 25
- #define TFALSE 26
- #define TNUMBER 27
- #define TSTRING 28
- #define TPLUS 29
- #define TMINUS 30
- #define TSTAR 31
- #define TEQUAL 32
- #define TNOTEQ 33
- #define TLE 34
- #define TLEEQ 35
- #define TGR 36
- #define TGREQ 37
- #define TLPAREN 38
- #define TRPAREN 39
- #define TLSQPAREN 40
- #define TRSQPAREN 41
- #define TASSIGN 42
- #define TDOT 43
- #define TCOMMA 44
- #define TCOLON 45
- #define TSEMI 46
- #define TREAD 47
- #define TWRITE 48
- #define TBREAK 49
- #define NUMOFTOKEN 49
- #define MAX_NUM_ATTR 32767
- #define KEYWORDS_SIZE 28
- struct KEY key [KEYWORDS_SIZE]

- FILE * `fp`
- int `num_attr`
- char `string_attr` [MAXSTRSIZE]
- void `error` (char *mes)
display an error message
- int `init_scan` (char *filename)
Initialization to begin scanning
- int `scan` (void)
Scan the file and return the token code
- int `get_linenum` (void)
Return the line number of the last token scanned
- int `end_scan` (void)
The process of finishing the scan

4.4.1 マクロ定義詳解

4.4.1.1 KEYWORDSIZE

```
#define KEYWORDSIZE 28
```

number of keywords

token-list.h の 122 行目に定義があります。

4.4.1.2 MAX_NUM_ATTR

```
#define MAX_NUM_ATTR 32767
```

maximum number of an unsigned integer

token-list.h の 118 行目に定義があります。

4.4.1.3 MAXSTRSIZE

```
#define MAXSTRSIZE 1024
```

maximum length of a string

token-list.h の 10 行目に定義があります。

4.4.1.4 NUMOFTOKEN

```
#define NUMOFTOKEN 49
```

number of tokens

token-list.h の 115 行目に定義があります。

4.4.1.5 TAND

```
#define TAND 19
```

and : Keyword

token-list.h の 51 行目に定義があります。

4.4.1.6 TARRAY

```
#define TARRAY 4
```

array : Keyword

token-list.h の 21 行目に定義があります。

4.4.1.7 TASSIGN

```
#define TASSIGN 42
```

:= : symbol

token-list.h の 97 行目に定義があります。

4.4.1.8 TBEGIN

```
#define TBEGIN 6
```

begin : Keyword

token-list.h の 25 行目に定義があります。

4.4.1.9 TBOOLEAN

```
#define TBOOLEAN 22
```

boolean : Keyword

token-list.h の 57 行目に定義があります。

4.4.1.10 TBREAK

```
#define TBREAK 49
```

break : Keyword

token-list.h の 111 行目に定義があります。

4.4.1.11 TCALL

```
#define TCALL 13
```

call : Keyword

token-list.h の 39 行目に定義があります。

4.4.1.12 TCHAR

```
#define TCHAR 20
```

char : Keyword

token-list.h の 53 行目に定義があります。

4.4.1.13 TCOLON

```
#define TCOLON 45
```

:: symbol

token-list.h の 103 行目に定義があります。

4.4.1.14 TCOMMA

```
#define TCOMMA 44
```

, : symbol

token-list.h の 101 行目に定義があります。

4.4.1.15 TDIV

```
#define TDIV 18
```

div : Keyword

token-list.h の 49 行目に定義があります。

4.4.1.16 TDO

```
#define TDO 15
```

do : Keyword

token-list.h の 43 行目に定義があります。

4.4.1.17 TDOT

```
#define TDOT 43
```

. : symbol

token-list.h の 99 行目に定義があります。

4.4.1.18 TELSE

```
#define TELSE 10
```

else : Keyword

token-list.h の 33 行目に定義があります。

4.4.1.19 TEND

```
#define TEND 7
```

end : Keyword

token-list.h の 27 行目に定義があります。

4.4.1.20 TEQUAL

```
#define TEQUAL 32
```

= : symbol

token-list.h の 77 行目に定義があります。

4.4.1.21 TFALSE

```
#define TFALSE 26
```

false : Keyword

token-list.h の 65 行目に定義があります。

4.4.1.22 TGR

```
#define TGR 36
```

: symbol

token-list.h の 85 行目に定義があります。

4.4.1.23 TGREQ

```
#define TGREQ 37
```

>= : symbol

token-list.h の 87 行目に定義があります。

4.4.1.24 TIF

```
#define TIF 8
```

if : Keyword

token-list.h の 29 行目に定義があります。

4.4.1.25 TINTEGER

```
#define TINTEGER 21
```

integer : Keyword

token-list.h の 55 行目に定義があります。

4.4.1.26 TLE

```
#define TLE 34
```

< : symbol

token-list.h の 81 行目に定義があります。

4.4.1.27 TLEEQ

```
#define TLEEQ 35
```

<= : symbol

token-list.h の 83 行目に定義があります。

4.4.1.28 TLPAREN

```
#define TLPAREN 38
```

(: symbol

token-list.h の 89 行目に定義があります。

4.4.1.29 TLSQPAREN

```
#define TLSQPAREN 40
```

[: symbol

token-list.h の 93 行目に定義があります。

4.4.1.30 TMINUS

```
#define TMINUS 30
```

• : symbol

token-list.h の 73 行目に定義があります。

4.4.1.31 TNAME

```
#define TNAME 1
```

Name : Alphabet { Alphabet | Digit }

token-list.h の 15 行目に定義があります。

4.4.1.32 TNOT

```
#define TNOT 16
```

not : Keyword

token-list.h の 45 行目に定義があります。

4.4.1.33 TNOTEQ

```
#define TNOTEQ 33
```

<> : symbol

token-list.h の 79 行目に定義があります。

4.4.1.34 TNUMBER

```
#define TNUMBER 27
```

unsigned integer

token-list.h の 67 行目に定義があります。

4.4.1.35 TOF

```
#define TOF 5
```

of : Keyword

token-list.h の 23 行目に定義があります。

4.4.1.36 TOR

```
#define TOR 17
```

or : Keyword

token-list.h の 47 行目に定義があります。

4.4.1.37 TPLUS

```
#define TPLUS 29
```

- : symbol

token-list.h の 71 行目に定義があります。

4.4.1.38 TPROCEDURE

```
#define TPROCEDURE 11
```

procedure : Keyword

token-list.h の 35 行目に定義があります。

4.4.1.39 TPROGRAM

```
#define TPROGRAM 2
```

program : Keyword

token-list.h の 17 行目に定義があります。

4.4.1.40 TREAD

```
#define TREAD 47
```

read : Keyword

token-list.h の 107 行目に定義があります。

4.4.1.41 TREADLN

```
#define TREADLN 23
```

readln : Keyword

token-list.h の 59 行目に定義があります。

4.4.1.42 TRETURN

```
#define TRETURN 12
```

return : Keyword

token-list.h の 37 行目に定義があります。

4.4.1.43 TRPAREN

```
#define TRPAREN 39
```

): symbol

token-list.h の 91 行目に定義があります。

4.4.1.44 TRSQPAREN

```
#define TRSQPAREN 41
```

] : symbol

token-list.h の 95 行目に定義があります。

4.4.1.45 TSEMI

```
#define TSEMI 46
```

:: symbol

token-list.h の 105 行目に定義があります。

4.4.1.46 TSTAR

```
#define TSTAR 31
```

- : symbol

token-list.h の 75 行目に定義があります。

4.4.1.47 TSTRING

```
#define TSTRING 28
```

String

token-list.h の 69 行目に定義があります。

4.4.1.48 TTHEN

```
#define TTHEN 9
```

then : Keyword

token-list.h の 31 行目に定義があります。

4.4.1.49 TTRUE

```
#define TTRUE 25
```

true : Keyword

token-list.h の 63 行目に定義があります。

4.4.1.50 TVAR

```
#define TVAR 3
```

var : Keyword

token-list.h の 19 行目に定義があります。

4.4.1.51 TWHILE

```
#define TWHILE 14
```

while : Keyword

token-list.h の 41 行目に定義があります。

4.4.1.52 TWRITE

```
#define TWRITE 48
```

write : Keyword

token-list.h の 109 行目に定義があります。

4.4.1.53 TWRITELN

```
#define TWRITELN 24
```

writeln : Keyword

token-list.h の 61 行目に定義があります。

4.4.2 関数詳解

4.4.2.1 end_scan()

```
int end_scan (
    void )
```

The process of finishing the scan

戻り値

int Returns 0 on success and -1 on failure.

scan.c の 123 行目に定義があります。

```
123     {
124         if (fclose(fp) == EOF) {
125             return -1;
126         }
127         return 0;
128     }
```

被呼び出し関係図:



4.4.2.2 error()

```
void error (
    char * mes )
```

display an error message

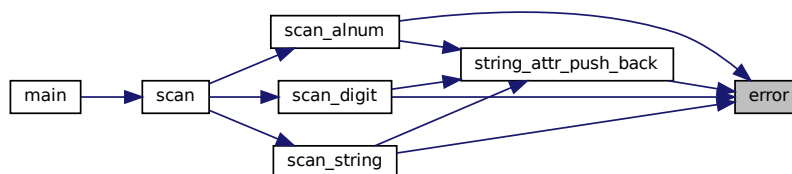
引数

in	mes	Error message
----	-----	---------------

token-list.c の 94 行目に定義があります。

```
94     {
95         fprintf(stderr, "\n ERROR: %s\n", mes);
96     }
```

被呼び出し関係図:



4.4.2.3 get_linenum()

```
int get_linenum (
    void )
```

Return the line number of the last token scanned

戻り値

int Return line number

scan.c の 108 行目に定義があります。

```
108     {
109         return token.linenum;
110     }
```

4.4.2.4 init_scan()

```
int init_scan (
    char * filename )
```

Initialization to begin scanning

引数

in	filename	File name to scan
----	----------	-------------------

戻り値

int Returns 0 on success and -1 on failure.

scan.c の 44 行目に定義があります。

```
44     {
```

```
45     if ((fp = fopen(filename, "r")) == NULL) {
46         return -1;
47     }
48
49     look_ahead();
50     look_ahead();
51
52     return 0;
53 }
```

呼び出し関係図:



被呼び出し関係図:



4.4.2.5 scan()

```
int scan (
    void )
```

Scan the file and return the token code

戻り値

int Returns token code on success and -1 on failure.

scan.c の 59 行目に定義があります。

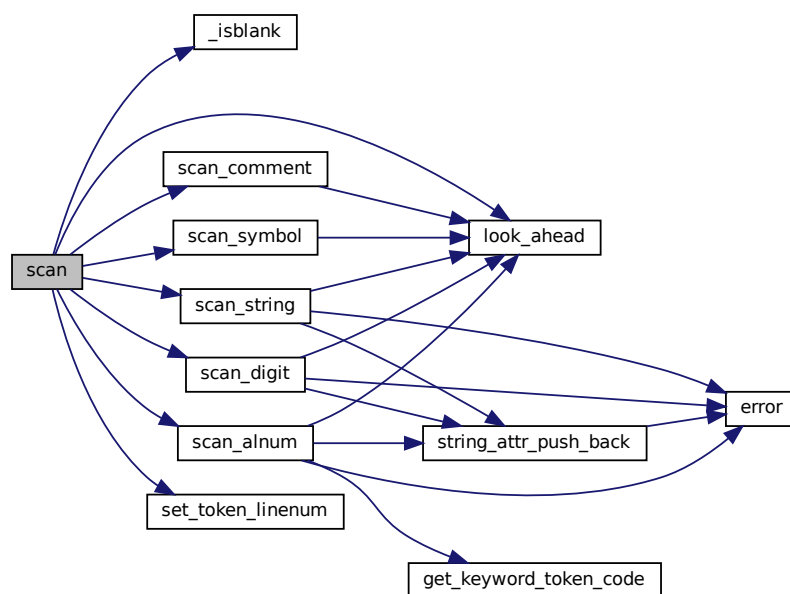
```
59     {
60         int token_code = -1;
61         while (1) {
62             if (current_char == EOF) { /* End Of File*/
63                 return -1;
64             } else if (current_char == '\r' || current_char == '\n') { /* End of Line */
65                 if (current_char == '\r') {
66                     if (next_char == '\n') {
67                         look_ahead();
68                     }
69                     look_ahead();
70                     linenum++;
71                 }
72             }
73         }
74     }
```

```

71         } else {
72             if (next_char == '\\r') {
73                 look_ahead();
74             }
75             look_ahead();
76             linenum++;
77         }
78     } else if (_isblank(current_char)) { /* Separator (Space or Tab) */
79         look_ahead();
80     } else if (!isprint(current_char)) { /* Not Graphic Character(0x20~0x7e) */
81         return -1;
82     } else if (isalpha(current_char)) { /* Name or Keyword */
83         token_code = scan_alnum();
84         break;
85     } else if (isdigit(current_char)) { /* Digit */
86         token_code = scan_digit();
87         break;
88     } else if (current_char == '\\') { /* String */
89         token_code = scan_string();
90         break;
91     } else if ((current_char == '/' && next_char == '*') || current_char == '{') { /* Comment */
92         if (scan_comment() == -1) {
93             break;
94         }
95     } else { /* Symbol */
96         token_code = scan_symbol();
97         break;
98     }
99 }
100 set_token_linenum();
101 return token_code;
102 }

```

呼び出し関係図:



被呼び出し関係図:



4.4.3 変数詳解

4.4.3.1 fp

```
FILE* fp [extern]
```

File pointer of the loaded file

scan.c の 9 行目に定義があります。

4.4.3.2 key

```
struct KEY key[KEYWORDSIZE]
```

4.4.3.3 num_attr

```
int num_attr [extern]
```

Scanned unsigned integer

scan.c の 11 行目に定義があります。

4.4.3.4 string_attr

```
char string_attr[MAXSTRSIZE] [extern]
```

Scanned string

scan.c の 13 行目に定義があります。

Index

- `_isblank`
 - `scan.c`, [12](#)
- `count`
 - `ID`, [5](#)
- `current_char`
 - `scan.c`, [24](#)
- `end_scan`
 - `scan.c`, [12](#)
 - `token-list.h`, [44](#)
- `error`
 - `token-list.c`, [26](#)
 - `token-list.h`, [44](#)
- `fp`
 - `scan.c`, [24](#)
 - `token-list.h`, [48](#)
- `get_keyword_token_code`
 - `scan.c`, [13](#)
- `get_linenum`
 - `scan.c`, [14](#)
 - `token-list.h`, [45](#)
- `ID`, [5](#)
 - `count`, [5](#)
 - `name`, [5](#)
 - `nextp`, [6](#)
- `id-list.c`, [7](#)
 - `id_countup`, [8](#)
 - `idroot`, [10](#)
 - `init_idtab`, [8](#)
 - `print_idtab`, [9](#)
 - `release_idtab`, [9](#)
 - `search_idtab`, [9](#)
- `id_countup`
 - `id-list.c`, [8](#)
- `idroot`
 - `id-list.c`, [10](#)
- `init_idtab`
 - `id-list.c`, [8](#)
- `init_scan`
 - `scan.c`, [14](#)
 - `token-list.h`, [45](#)
- `KEY`, [6](#)
 - `keytoken`, [6](#)
 - `keyword`, [6](#)
- `key`
 - `token-list.c`, [29](#)
 - `token-list.h`, [48](#)
- `keytoken`
 - `KEY`, [6](#)
- `keyword`
 - `KEY`, [6](#)
- `KEYWORDSIZE`
 - `token-list.h`, [33](#)
- `linenum`
 - `scan.c`, [25](#)
- `look_ahead`
 - `scan.c`, [15](#)
- `main`
 - `token-list.c`, [28](#)
- `MAX_NUM_ATTR`
 - `token-list.h`, [33](#)
- `MAXSTRSIZE`
 - `token-list.h`, [33](#)
- `name`
 - `ID`, [5](#)
- `next_char`
 - `scan.c`, [25](#)
- `nextp`
 - `ID`, [6](#)
- `num_attr`
 - `scan.c`, [25](#)
 - `token-list.h`, [48](#)
- `NUMOFTOKEN`
 - `token-list.h`, [33](#)
- `numtoken`
 - `token-list.c`, [30](#)
- `print_idtab`
 - `id-list.c`, [9](#)
- `release_idtab`
 - `id-list.c`, [9](#)
- `scan`
 - `scan.c`, [16](#)
 - `token-list.h`, [46](#)
- `scan.c`, [10](#)
 - `_isblank`, [12](#)
 - `current_char`, [24](#)
 - `end_scan`, [12](#)
 - `fp`, [24](#)
 - `get_keyword_token_code`, [13](#)
 - `get_linenum`, [14](#)
 - `init_scan`, [14](#)

- linenum, 25
- look_ahead, 15
- next_char, 25
- num_attr, 25
- scan, 16
- scan_alnum, 17
- scan_comment, 18
- scan_digit, 19
- scan_string, 20
- scan_symbol, 21
- set_token_linenum, 23
- string_attr, 25
- string_attr_push_back, 23
- token_linenum, 25
- scan_alnum
 - scan.c, 17
- scan_comment
 - scan.c, 18
- scan_digit
 - scan.c, 19
- scan_string
 - scan.c, 20
- scan_symbol
 - scan.c, 21
- search_idtab
 - id-list.c, 9
- set_token_linenum
 - scan.c, 23
- string_attr
 - scan.c, 25
 - token-list.h, 48
- string_attr_push_back
 - scan.c, 23
- TAND
 - token-list.h, 34
- TARRAY
 - token-list.h, 34
- TASSIGN
 - token-list.h, 34
- TBEGIN
 - token-list.h, 34
- TBOOLEAN
 - token-list.h, 34
- TBREAK
 - token-list.h, 35
- TCALL
 - token-list.h, 35
- TCHAR
 - token-list.h, 35
- TCOLON
 - token-list.h, 35
- TCOMMA
 - token-list.h, 35
- TDIV
 - token-list.h, 36
- TDO
 - token-list.h, 36
- TDOT
 - token-list.h, 36
- TELSE
 - token-list.h, 36
- TEND
 - token-list.h, 36
- TEQUAL
 - token-list.h, 37
- TFALSE
 - token-list.h, 37
- TGR
 - token-list.h, 37
- TGREQ
 - token-list.h, 37
- TIF
 - token-list.h, 37
- TINTEGER
 - token-list.h, 38
- TLE
 - token-list.h, 38
- TLEEQ
 - token-list.h, 38
- TLPAREN
 - token-list.h, 38
- TLSQPAREN
 - token-list.h, 38
- TMINUS
 - token-list.h, 39
- TNAME
 - token-list.h, 39
- TNOT
 - token-list.h, 39
- TNOTEQ
 - token-list.h, 39
- TNUMBER
 - token-list.h, 39
- TOF
 - token-list.h, 40
- token-list.c, 26
 - error, 26
 - key, 29
 - main, 28
 - numtoken, 30
 - tokenstr, 30
- token-list.h, 31
 - end_scan, 44
 - error, 44
 - fp, 48
 - get_linenum, 45
 - init_scan, 45
 - key, 48
 - KEYWORDSIZE, 33
 - MAX_NUM_ATTR, 33
 - MAXSTRSIZE, 33
 - num_attr, 48
 - NUMOFTOKEN, 33
 - scan, 46
 - string_attr, 48
 - TAND, 34

- TARRAY, [34](#)
 - TASSIGN, [34](#)
 - TBEGIN, [34](#)
 - TBOOLEAN, [34](#)
 - TBREAK, [35](#)
 - TCALL, [35](#)
 - TCHAR, [35](#)
 - TCOLON, [35](#)
 - TCOMMA, [35](#)
 - TDIV, [36](#)
 - TDO, [36](#)
 - TDOT, [36](#)
 - TELSE, [36](#)
 - TEND, [36](#)
 - TEQUAL, [37](#)
 - TFALSE, [37](#)
 - TGR, [37](#)
 - TGREQ, [37](#)
 - TIF, [37](#)
 - TINTEGER, [38](#)
 - TLE, [38](#)
 - TLEEQ, [38](#)
 - TLPAREN, [38](#)
 - TLSPAREN, [38](#)
 - TMINUS, [39](#)
 - TNAME, [39](#)
 - TNOT, [39](#)
 - TNOTEQ, [39](#)
 - TNUMBER, [39](#)
 - TOF, [40](#)
 - TOR, [40](#)
 - TPLUS, [40](#)
 - TPROCEDURE, [40](#)
 - TPROGRAM, [40](#)
 - token-list.h, [40](#)
 - TREAD, [41](#)
 - token-list.h, [41](#)
 - TREADLN, [41](#)
 - token-list.h, [41](#)
 - TRETURN, [41](#)
 - token-list.h, [41](#)
 - TRPAREN, [41](#)
 - token-list.h, [41](#)
 - TRSQPAREN, [41](#)
 - token-list.h, [41](#)
 - TSEMI, [42](#)
 - token-list.h, [42](#)
 - TSTAR, [42](#)
 - token-list.h, [42](#)
 - TSTRING, [42](#)
 - token-list.h, [42](#)
 - TTHEN, [42](#)
 - token-list.h, [42](#)
 - TTRUE, [42](#)
 - token-list.h, [42](#)
 - TVAR, [43](#)
 - token-list.h, [43](#)
 - TWHILE, [43](#)
 - token-list.h, [43](#)
 - TWRITE, [43](#)
 - token-list.h, [43](#)
 - TWRITELN, [43](#)
 - token-list.h, [43](#)
- token_linenum
- scan.c, [25](#)
- tokenstr
- token-list.c, [30](#)
- TOR
- token-list.h, [40](#)
- TPLUS
- token-list.h, [40](#)
- TPROCEDURE
- token-list.h, [40](#)