

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	11
4.2.1.1 _isblank()	12
4.2.1.2 end_scan()	12
4.2.1.3 get_keyword_token_code()	13
4.2.1.4 get_linenum()	13
4.2.1.5 init_scan()	13
4.2.1.6 look_ahead()	14
4.2.1.7 scan()	15
4.2.1.8 scan_alnum()	16
4.2.1.9 scan_comment()	17
4.2.1.10 scan_digit()	18
4.2.1.11 scan_string()	19
4.2.1.12 scan_symbol()	20

4.2.1.13 set_token_linenum()	21
4.2.1.14 string_attr_push_back()	22
4.2.2 変数詳解	22
4.2.2.1 current_char	22
4.2.2.2 fp	23
4.2.2.3 linenum	23
4.2.2.4 next_char	23
4.2.2.5 num_attr	23
4.2.2.6 string_attr	23
4.2.2.7 token_linenum	23
4.3 token-list.c ファイル	24
4.3.1 関数詳解	24
4.3.1.1 error()	24
4.3.1.2 main()	25
4.3.2 変数詳解	26
4.3.2.1 key	26
4.3.2.2 numtoken	27
4.3.2.3 tokenstr	27
4.4 token-list.h ファイル	27
4.4.1 マクロ定義詳解	29
4.4.1.1 KEYWORDSIZE	29
4.4.1.2 MAX_NUM_ATTR	30
4.4.1.3 MAXSTRSIZE	30
4.4.1.4 NUMOFTOKEN	30
4.4.1.5 TAND	30
4.4.1.6 TARRAY	30
4.4.1.7 TASSIGN	30
4.4.1.8 TBEGIN	31
4.4.1.9 TBOOLEAN	31
4.4.1.10 TBREAK	31
4.4.1.11 TCALL	31
4.4.1.12 TCHAR	31
4.4.1.13 TCOLON	31
4.4.1.14 TCOMMA	32
4.4.1.15 TDIV	32
4.4.1.16 TDO	32
4.4.1.17 TDOT	32
4.4.1.18 TELSE	32
4.4.1.19 TEND	32
4.4.1.20 TEQUAL	33
4.4.1.21 TFALSE	33
4.4.1.22 TGR	33

4.4.1.23 TGREQ	33
4.4.1.24 TIF	33
4.4.1.25 TINTEGER	33
4.4.1.26 TLE	34
4.4.1.27 TLEEQ	34
4.4.1.28 TLPAREN	34
4.4.1.29 TLSQPAREN	34
4.4.1.30 TMINUS	34
4.4.1.31 TNAME	34
4.4.1.32 TNOT	35
4.4.1.33 TNOTEQ	35
4.4.1.34 TNUMBER	35
4.4.1.35 TOF	35
4.4.1.36 TOR	35
4.4.1.37 TPLUS	35
4.4.1.38 TPROCEDURE	36
4.4.1.39 TPROGRAM	36
4.4.1.40 TREAD	36
4.4.1.41 TREADLN	36
4.4.1.42 TRETURN	36
4.4.1.43 TRPAREN	36
4.4.1.44 TRSQPAREN	37
4.4.1.45 TSEMI	37
4.4.1.46 TSTAR	37
4.4.1.47 TSTRING	37
4.4.1.48 TTHEN	37
4.4.1.49 TTRUE	37
4.4.1.50 TVAR	38
4.4.1.51 TWHILE	38
4.4.1.52 TWRITE	38
4.4.1.53 TWRITELN	38
4.4.2 関数詳解	38
4.4.2.1 end_scan()	38
4.4.2.2 error()	39
4.4.2.3 get_linenum()	39
4.4.2.4 init_scan()	39
4.4.2.5 scan()	40
4.4.3 変数詳解	41
4.4.3.1 fp	41
4.4.3.2 key	42
4.4.3.3 num_attr	42
4.4.3.4 string_attr	42

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

Chapter 2

ファイル索引

2.1 ファイル一覧

ファイル一覧です。

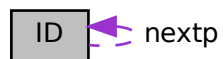
id-list.c	7
scan.c	10
token-list.c	24
token-list.h	27

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

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

フィールド

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

3.2.1 詳解

token-list.h の 70 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
```

token-list.h の 72 行目に定義があります。

3.2.2.2 keyword

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

token-list.h の 71 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](#)

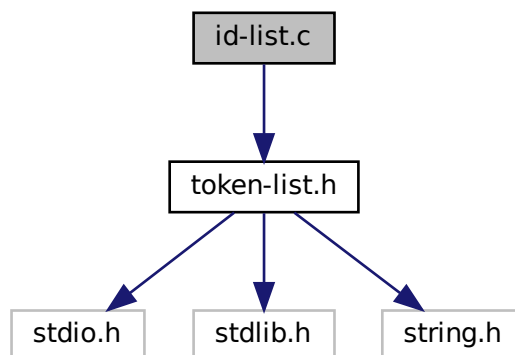
Chapter 4

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:



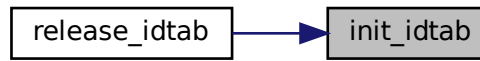
4.1.1.2 init_idtab()

```
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
```

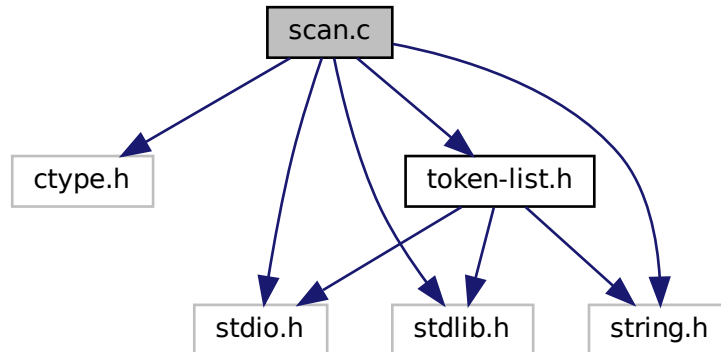
4.2 scan.c ファイル

```
#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:



関数

- static int `_isblank` (int c)
- static int `scan_alnum` ()
- static int `scan_digit` ()
- static int `scan_string` ()
- static int `scan_comment` ()
- static int `scan_symbol` ()
- static int `get_keyword_token_code` (char *token)
- static void `look_ahead` ()
- static int `string_attr_push_back` (const char c)
- int `init_scan` (char *filename)
- int `scan` (void)
- int `get_linenum` (void)
- void `set_token_linenum` (void)
- int `end_scan` (void)

変数

- FILE * `fp`
- char `string_attr` [MAXSTRSIZE]
- static int `current_char`
- static int `next_char` = '\0'
- static int `linenum` = 1
- static int `token_linenum` = 0
- int `num_attr` = 0

4.2.1 関数詳解

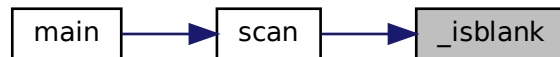
4.2.1.1 _isblank()

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

scan.c の 96 行目に定義があります。

```
96      {  
97      if (c == ' ' || c == '\t') {  
98          return 1;  
99      } else {  
100         return 0;  
101     }  
102 }
```

被呼び出し関係図:



4.2.1.2 end_scan()

```
int endscan (  
    void )
```

scan.c の 89 行目に定義があります。

```
89      {  
90      if (fclose(fp) == EOF) {  
91          return -1;  
92      }  
93      return 0;  
94 }
```

被呼び出し関係図:



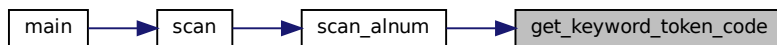
4.2.1.3 get_keyword_token_code()

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

scan.c の 245 行目に定義があります。

```
245                                     {
246     int index;
247     /* TODO: binary search */
248     for (index = 0; index < KEYWORDSIZE; index++) {
249         if (strcmp(token, key[index].keyword) == 0) {
250             /* This token is Keyword */
251             return key[index].keytoken;
252         }
253     }
254     /* This token is NAME*/
255     return TNAME;
256 }
```

被呼び出し関係図:



4.2.1.4 get_linenum()

```
int get_linenum (
    void )
```

scan.c の 81 行目に定義があります。

```
81     {
82     return token.linenum;
83 }
```

4.2.1.5 init_scan()

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

scan.c の 26 行目に定義があります。

```
26     {
27     if ((fp = fopen(filename, "r")) == NULL) {
28         return -1;
29     }
30
31     look_ahead();
32     look_ahead();
33
34     return 0;
35 }
```

呼び出し関係図:



被呼び出し関係図:



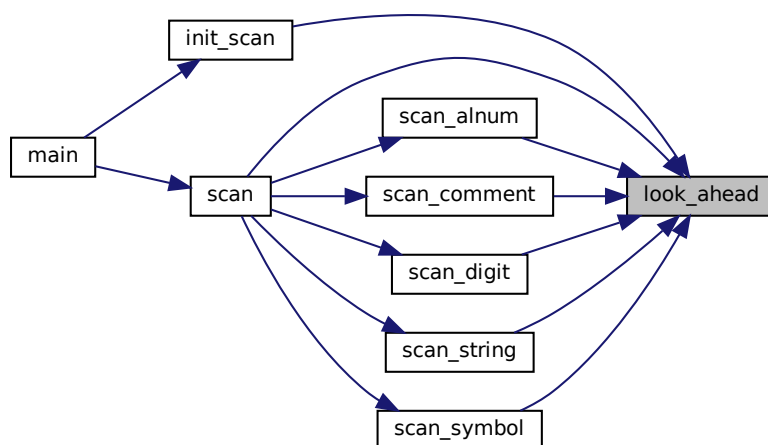
4.2.1.6 look_ahead()

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

scan.c の 270 行目に定義があります。

```
270     {  
271         current_char = next_char;  
272         next_char = fgetc(fp);  
273         return;  
274     }
```

被呼び出し関係図:



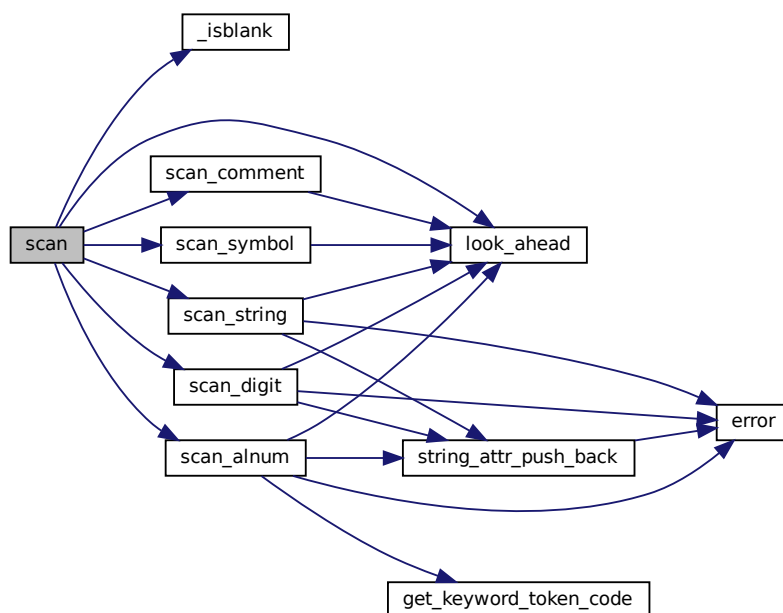
4.2.1.7 scan()

```
int scan (
    void )
```

scan.c の 37 行目に定義があります。

```
37     {
38         int token_code = -1;
39         while (1) {
40             if (current_char == EOF) { /* End Of File*/
41                 return -1;
42             } else if (current_char == '\r' || current_char == '\n') { /* End of Line */
43                 if (current_char == '\r') {
44                     if (next_char == '\n') {
45                         look_ahead();
46                     }
47                     look_ahead();
48                     linenum++;
49                 } else {
50                     if (next_char == '\r') {
51                         look_ahead();
52                     }
53                     look_ahead();
54                     linenum++;
55                 }
56             } else if (!isblank(current_char)) { /* Separator (Space or Tab) */
57                 look_ahead();
58             } else if (!isprint(current_char)) { /* Not Graphic Character(0x20~0x7e) */
59                 return -1;
60             } else if (isalpha(current_char)) { /* Name or Keyword */
61                 token_code = scan_alnum();
62                 break;
63             } else if (isdigit(current_char)) { /* Digit */
64                 token_code = scan_digit();
65                 break;
66             } else if (current_char == '\'') { /* String */
67                 token_code = scan_string();
68                 break;
69             } else if ((current_char == '/' && next_char == '*') || current_char == '{') { /* Comment */
70                 if (scan_comment() == -1) {
71                     break;
72                 }
73             } else { /* Symbol */
74                 token_code = scan_symbol();
75                 break;
76             }
77         }
78         return token_code;
79     }
```

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.c の 104 行目に定義があります。

```

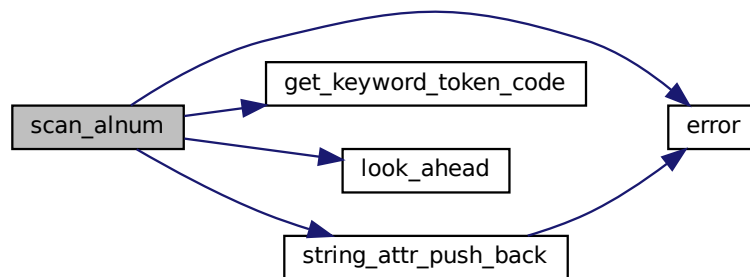
104     {
105     memset(string_attr, '\0', sizeof(string_attr));
106     string_attr[0] = current_char;
107
108     look_ahead();
109     while (isalnum(current_char)) {
110         if (string_attr.push_back(current_char) == -1) {
111             error("function scan_alnum()");
112             return -1;
113         }
114         look_ahead();
115     }
  
```

```

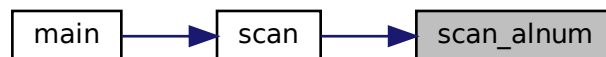
116     return get_keyword_token_code(string_attr);
117 }

```

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.c の 164 行目に定義があります。

```

164     {
165     if (current_char == '/' && next_char == '*') {
166         look_ahead();
167         look_ahead();
168         while (current_char != EOF) {
169             if (current_char == '*' && next_char == '/') {
170                 look_ahead();
171                 look_ahead();
172                 return 0;
173             }
174             look_ahead();
175         }
176     } else if (current_char == '{') {
177         look_ahead();
178         while (current_char != EOF) {
179             if (current_char == '}') {
180                 look_ahead();
181                 return 0;
182             }
183             look_ahead();
184         }
185     }

```

```
186     /* EOF */
187     return -1;
188 }
```

呼び出し関係図:



被呼び出し関係図:



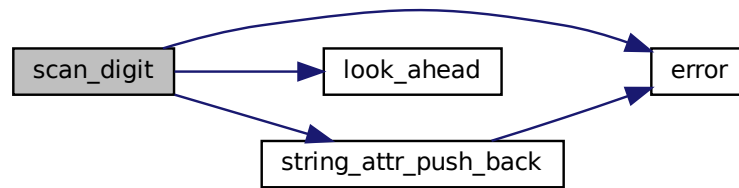
4.2.1.10 scan_digit()

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

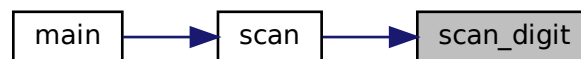
scan.c の 119 行目に定義があります。

```
119     {
120         int num = current_char - '0';
121
122         memset(string_attr, '\0', sizeof(string_attr));
123         string_attr[0] = current_char;
124
125         look_ahead();
126         while (isdigit(current_char)) {
127             if (string_attr.push_back(current_char) == -1) {
128                 error("function scan_digit()");
129                 return -1;
130             }
131             num *= 10;
132             num += current_char - '0';
133             look_ahead();
134         }
135         if (num <= MAX_NUM_ATTR) {
136             num_attr = num;
137             return TNUMBER;
138         }
139
140         return -1;
141     }
```


呼び出し関係図:



被呼び出し関係図:



4.2.1.11 scan_string()

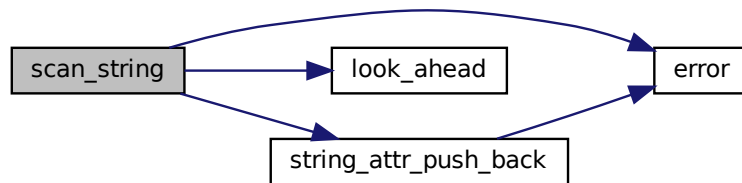
```
static int scan_string ( ) [static]
```

scan.c の 143 行目に定義があります。

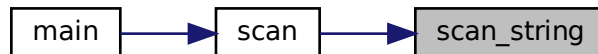
```

143     {
144     memset(string_attr, '\0', sizeof(string_attr));
145     look_ahead();
146
147     while ((!(current_char == '\\' || next_char == '\\')) {
148     if (!isprint(current_char)) {
149     error("function scan_string()");
150     fprintf(stderr, "[%c]0x%x is not graphic character.\n", current_char, current_char);
151     return -1;
152     }
153     if (string_attr_push_back(current_char) == -1) {
154     error("function scan_digit()");
155     return -1;
156     }
157     look_ahead();
158     }
159     look_ahead(); /* read '\\' */
160
161     return TSTRING;
162 }
  
```

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.c の 190 行目に定義があります。

```

190     {
191     char symbol = current_char;
192     look_ahead();
193     switch (symbol) {
194     case '+':
195         return TPLUS;
196     case '-':
197         return TMINUS;
198     case '*':
199         return TSTAR;
200     case '=':
201         return TEQUAL;
202     case '<':
203         if (current_char == '>') {
204             look_ahead();
205             return TNOTEQ;
206         } else if (current_char == '=') {
207             look_ahead();
208             return TLEEQ;
209         } else {
210             return TLE;
211         }
212     case '>':
213         if (current_char == '=') {
214             look_ahead();
215             return TGREQ;
216         } else {
217             return TGR;
218         }
219     case '(':
  
```

```

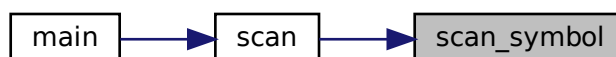
220         return TLPAREN;
221     case ')':
222         return TRPAREN;
223     case '[':
224         return TLSQPAREN;
225     case ']':
226         return TRSQPAREN;
227     case ':':
228         if (current_char == '=') {
229             look_ahead();
230             return TASSIGN;
231         } else {
232             return TCOLON;
233         }
234     case '.':
235         return TDOT;
236     case ',':
237         return TCOMMA;
238     case ';':
239         return TSEMI;
240     default:
241         return -1;
242     }
243 }

```

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 )

```

scan.c の 85 行目に定義があります。

```

85     {
86         token_linenum = linenum;
87     }

```

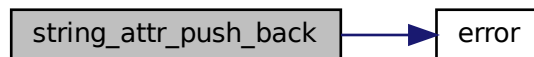
4.2.1.14 string_attr_push_back()

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

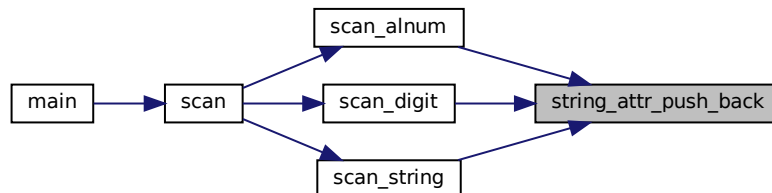
scan.c の 258 行目に定義があります。

```
258 {
259     int len = strlen(string_attr);
260     if (len < MAXSTRSIZE - 1) {
261         string_attr[len] = c;
262         return 0;
263     } else {
264         /* Buffer Overflow */
265         error("string_attr: Buffer Overflow.");
266         return -1;
267     }
268 }
```

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]
```

scan.c の 10 行目に定義があります。

4.2.2.2 fp

```
FILE* fp
```

scan.c の 8 行目に定義があります。

4.2.2.3 linenum

```
int linenum = 1 [static]
```

scan.c の 22 行目に定義があります。

4.2.2.4 next_char

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

scan.c の 11 行目に定義があります。

4.2.2.5 num_attr

```
int num_attr = 0
```

scan.c の 24 行目に定義があります。

4.2.2.6 string_attr

```
char string_attr[MAXSTRSIZE]
```

scan.c の 9 行目に定義があります。

4.2.2.7 token_linenum

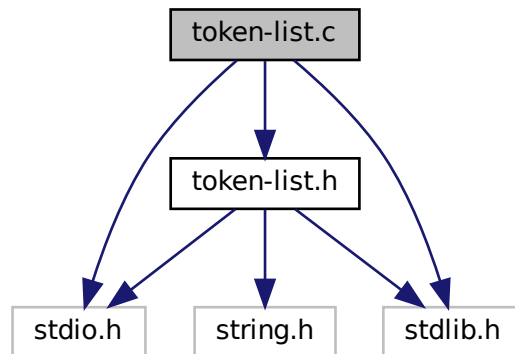
```
int token_linenum = 0 [static]
```

scan.c の 23 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[])
- void `error` (char *mes)

変数

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

4.3.1 関数詳解

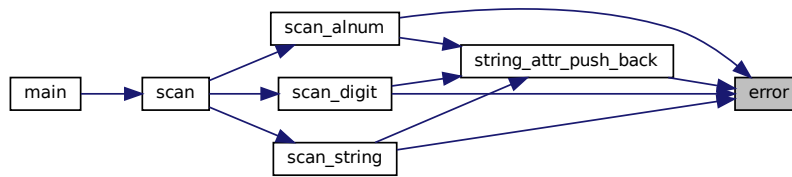
4.3.1.1 error()

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

token-list.c の 87 行目に定義があります。

```
87     {
88         fprintf(stderr, "\n ERROR: %s\n", mes);
89         /* end_scan(); */
90     }
```

被呼び出し関係図:



4.3.1.2 main()

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

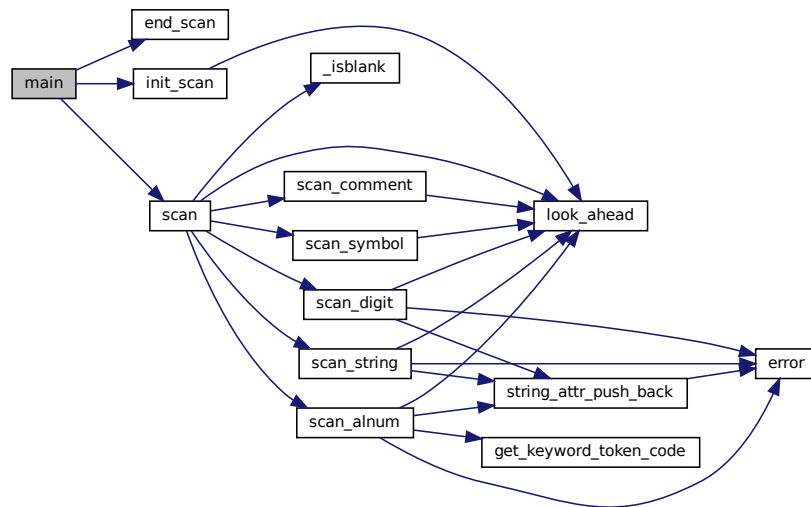
token-list.c の 54 行目に定義があります。

```

54     {
55         int token, index;
56
57         if (nc < 2) {
58             fprintf(stderr, "File name id not given.\n");
59             return EXIT_FAILURE;
60         }
61         if (init_scan(np[1]) < 0) {
62             fprintf(stderr, "File %s can not open.\n", np[1]);
63             return EXIT_FAILURE;
64         }
65
66         memset(numtoken, 0, sizeof(numtoken));
67
68         while ((token = scan()) >= 0) {
69             /* 作成する部分: トークンをカウントする */
70             numtoken[token]++;
71         }
72
73         if (end_scan() < 0) {
74             fprintf(stderr, "File %s can not close.\n", np[1]);
75             return EXIT_FAILURE;
76         }
77         /* 作成する部分: カウントした結果を出力する */
78         for (index = 0; index < NUMOFTOKEN + 1; index++) {
79             if (numtoken[index] > 0) {
80                 fprintf(stdout, "%10s: %5d\n", tokenstr[index], numtoken[index]);
81             }
82         }
83
84         return 0;
85     }

```

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},
    {"end", TEND},
    {"false", TFALSE},
    {"if", TIF},
    {"integer", TINTEGER},
    {"not", TNOT},
    {"of", TOF},
    {"or", TOR},
    {"procedure", TPROCEDURE},
    {"program", TPROGRAM},
    {"read", TREAD},
    {"readln", TREADLN},
    {"return", TRETURN},
    {"then", TTHEN},
    {"true", TTRUE},
    {"var", TVAR},
    {"while", TWHILE},
    {"write", TWRITE},
    {"writeln", TWriteln}}

```

token-list.c の 1 行目に定義があります。

4.3.2.2 numtoken

```
int numtoken[NUMOFTOKEN+1]
```

token-list.c の 38 行目に定義があります。

4.3.2.3 tokenstr

```
char* tokenstr[NUMOFTOKEN+1]
```

初期値:

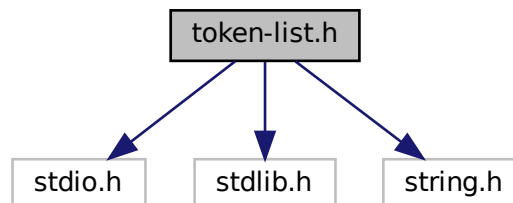
```
= {  
    "", "NAME", "program", "var", "array", "of", "begin",  
    "end", "if", "then", "else", "procedure", "return", "call",  
    "while", "do", "not", "or", "div", "and", "char",  
    "integer", "boolean", "readln", "writeln", "true", "false", "NUMBER",  
    "STRING", "+", "-", "*", "=", "<>", "<",  
    "<=", ">", ">=", "(", ")", "[", "]",  
    ":", ":", ":", ":", ":", ":", ":", ":", ":", ":", ":", ":", ":", ":", ":",  
    "break"}  
}
```

token-list.c の 41 行目に定義があります。

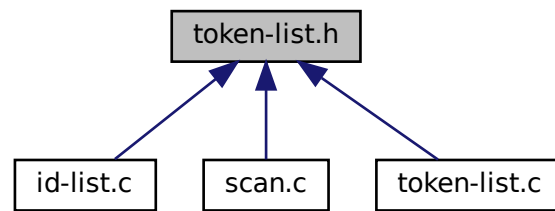
4.4 token-list.h ファイル

```
#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](#)

マクロ定義

- #define [MAXSTRSIZE](#) 1024
- #define [TNAME](#) 1 /* Name : Alphabet { Alphabet | Digit } */
- #define [TPROGRAM](#) 2 /* program : Keyword */
- #define [TVAR](#) 3 /* var : Keyword */
- #define [TARRAY](#) 4 /* array : Keyword */
- #define [TOF](#) 5 /* of : Keyword */
- #define [TBEGIN](#) 6 /* begin : Keyword */
- #define [TEND](#) 7 /* end : Keyword */
- #define [TIF](#) 8 /* if : Keyword */
- #define [TTHEN](#) 9 /* then : Keyword */
- #define [TELSE](#) 10 /* else : Keyword */
- #define [TPROCEDURE](#) 11 /* procedure : Keyword */
- #define [TRETURN](#) 12 /* return : Keyword */
- #define [TCALL](#) 13 /* call : Keyword */
- #define [TWHILE](#) 14 /* while : Keyword */
- #define [TDO](#) 15 /* do : Keyword */
- #define [TNOT](#) 16 /* not : Keyword */
- #define [TOR](#) 17 /* or : Keyword */
- #define [TDIV](#) 18 /* div : Keyword */
- #define [TAND](#) 19 /* and : Keyword */
- #define [TCHAR](#) 20 /* char : Keyword */
- #define [TINTEGER](#) 21 /* integer : Keyword */
- #define [TBOOLEAN](#) 22 /* boolean : Keyword */
- #define [TREADLN](#) 23 /* readln : Keyword */
- #define [TWRITELN](#) 24 /* writeln : Keyword */
- #define [TTRUE](#) 25 /* true : Keyword */
- #define [TFALSE](#) 26 /* false : Keyword */
- #define [TNUMBER](#) 27 /* unsigned integer */
- #define [TSTRING](#) 28 /* String */
- #define [TPLUS](#) 29 /* + : symbol */

- `#define TMINUS 30 /* - : symbol */`
- `#define TSTAR 31 /* * : symbol */`
- `#define TEQUAL 32 /* = : symbol */`
- `#define TNOTEQ 33 /* <> : symbol */`
- `#define TLE 34 /* < : symbol */`
- `#define TLEEQ 35 /* <= : symbol */`
- `#define TGR 36 /* > : symbol */`
- `#define TGREQ 37 /* >= : symbol */`
- `#define TLPAREN 38 /* (: symbol */`
- `#define TRPAREN 39 /*) : symbol */`
- `#define TLSQPAREN 40 /* [: symbol */`
- `#define TRSQPAREN 41 /*] : symbol */`
- `#define TASSIGN 42 /* := : symbol */`
- `#define TDOT 43 /* . : symbol */`
- `#define TCOMMA 44 /* , : symbol */`
- `#define TCOLON 45 /* : : symbol */`
- `#define TSEMI 46 /* ; : symbol */`
- `#define TREAD 47 /* read : Keyword */`
- `#define TWRITE 48 /* write : Keyword */`
- `#define TBREAK 49 /* break : Keyword */`
- `#define NUMOFTOKEN 49`
- `#define MAX_NUM_ATTR 32767`
- `#define KEYWORDSIZE 28`

関数

- `void error (char *mes)`
- `int init_scan (char *filename)`
- `int scan (void)`
- `int get_linenum (void)`
- `int end_scan (void)`

変数

- `struct KEY key [KEYWORDSIZE]`
- `FILE * fp`
- `int num_attr`
- `char string_attr [MAXSTRSIZE]`

4.4.1 マクロ定義詳解

4.4.1.1 KEYWORDSIZE

```
#define KEYWORDSIZE 28
```

token-list.h の 68 行目に定義があります。

4.4.1.2 MAX_NUM_ATTR

```
#define MAX_NUM_ATTR 32767
```

token-list.h の 64 行目に定義があります。

4.4.1.3 MAXSTRSIZE

```
#define MAXSTRSIZE 1024
```

token-list.h の 9 行目に定義があります。

4.4.1.4 NUMOFTOKEN

```
#define NUMOFTOKEN 49
```

token-list.h の 62 行目に定義があります。

4.4.1.5 TAND

```
#define TAND 19 /* and : Keyword */
```

token-list.h の 30 行目に定義があります。

4.4.1.6 TARRAY

```
#define TARRAY 4 /* array : Keyword */
```

token-list.h の 15 行目に定義があります。

4.4.1.7 TASSIGN

```
#define TASSIGN 42 /* := : symbol */
```

token-list.h の 53 行目に定義があります。

4.4.1.8 TBEGIN

```
#define TBEGIN 6 /* begin : Keyword */
```

token-list.h の 17 行目に定義があります。

4.4.1.9 TBOOLEAN

```
#define TBOOLEAN 22 /* boolean : Keyword */
```

token-list.h の 33 行目に定義があります。

4.4.1.10 TBREAK

```
#define TBREAK 49 /* break : Keyword */
```

token-list.h の 60 行目に定義があります。

4.4.1.11 TCALL

```
#define TCALL 13 /* call : Keyword */
```

token-list.h の 24 行目に定義があります。

4.4.1.12 TCHAR

```
#define TCHAR 20 /* char : Keyword */
```

token-list.h の 31 行目に定義があります。

4.4.1.13 TCOLON

```
#define TCOLON 45 /* : : symbol */
```

token-list.h の 56 行目に定義があります。

4.4.1.14 TCOMMA

```
#define TCOMMA 44 /* , : symbol */
```

token-list.h の 55 行目に定義があります。

4.4.1.15 TDIV

```
#define TDIV 18 /* div : Keyword */
```

token-list.h の 29 行目に定義があります。

4.4.1.16 TDO

```
#define TDO 15 /* do : Keyword */
```

token-list.h の 26 行目に定義があります。

4.4.1.17 TDOT

```
#define TDOT 43 /* . : symbol */
```

token-list.h の 54 行目に定義があります。

4.4.1.18 TELSE

```
#define TELSE 10 /* else : Keyword */
```

token-list.h の 21 行目に定義があります。

4.4.1.19 TEND

```
#define TEND 7 /* end : Keyword */
```

token-list.h の 18 行目に定義があります。

4.4.1.20 TEQUAL

```
#define TEQUAL 32 /* = : symbol */
```

token-list.h の 43 行目に定義があります。

4.4.1.21 TFALSE

```
#define TFALSE 26 /* false : Keyword */
```

token-list.h の 37 行目に定義があります。

4.4.1.22 TGR

```
#define TGR 36 /* > : symbol */
```

token-list.h の 47 行目に定義があります。

4.4.1.23 TGREQ

```
#define TGREQ 37 /* >= : symbol */
```

token-list.h の 48 行目に定義があります。

4.4.1.24 TIF

```
#define TIF 8 /* if : Keyword */
```

token-list.h の 19 行目に定義があります。

4.4.1.25 TINTEGER

```
#define TINTEGER 21 /* integer : Keyword */
```

token-list.h の 32 行目に定義があります。

4.4.1.26 TLE

```
#define TLE 34 /* < : symbol */
```

token-list.h の 45 行目に定義があります。

4.4.1.27 TLEEQ

```
#define TLEEQ 35 /* <= : symbol */
```

token-list.h の 46 行目に定義があります。

4.4.1.28 TLPAREN

```
#define TLPAREN 38 /* ( : symbol */
```

token-list.h の 49 行目に定義があります。

4.4.1.29 TLSQPAREN

```
#define TLSQPAREN 40 /* [ : symbol */
```

token-list.h の 51 行目に定義があります。

4.4.1.30 TMINUS

```
#define TMINUS 30 /* - : symbol */
```

token-list.h の 41 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 12 行目に定義があります。

4.4.1.32 TNOT

```
#define TNOT 16 /* not : Keyword */
```

token-list.h の 27 行目に定義があります。

4.4.1.33 TNOTEQ

```
#define TNOTEQ 33 /* <> : symbol */
```

token-list.h の 44 行目に定義があります。

4.4.1.34 TNUMBER

```
#define TNUMBER 27 /* unsigned integer */
```

token-list.h の 38 行目に定義があります。

4.4.1.35 TOF

```
#define TOF 5 /* of : Keyword */
```

token-list.h の 16 行目に定義があります。

4.4.1.36 TOR

```
#define TOR 17 /* or : Keyword */
```

token-list.h の 28 行目に定義があります。

4.4.1.37 TPLUS

```
#define TPLUS 29 /* + : symbol */
```

token-list.h の 40 行目に定義があります。

4.4.1.38 TPROCEDURE

```
#define TPROCEDURE 11 /* procedure : Keyword */
```

token-list.h の 22 行目に定義があります。

4.4.1.39 TPROGRAM

```
#define TPROGRAM 2 /* program : Keyword */
```

token-list.h の 13 行目に定義があります。

4.4.1.40 TREAD

```
#define TREAD 47 /* read : Keyword */
```

token-list.h の 58 行目に定義があります。

4.4.1.41 TREADLN

```
#define TREADLN 23 /* readln : Keyword */
```

token-list.h の 34 行目に定義があります。

4.4.1.42 TRETURN

```
#define TRETURN 12 /* return : Keyword */
```

token-list.h の 23 行目に定義があります。

4.4.1.43 TRPAREN

```
#define TRPAREN 39 /* ) : symbol */
```

token-list.h の 50 行目に定義があります。

4.4.1.44 TRSQPAREN

```
#define TRSQPAREN 41 /* ] : symbol */
```

token-list.h の 52 行目に定義があります。

4.4.1.45 TSEMI

```
#define TSEMI 46 /* ; : symbol */
```

token-list.h の 57 行目に定義があります。

4.4.1.46 TSTAR

```
#define TSTAR 31 /* * : symbol */
```

token-list.h の 42 行目に定義があります。

4.4.1.47 TSTRING

```
#define TSTRING 28 /* String */
```

token-list.h の 39 行目に定義があります。

4.4.1.48 TTHEN

```
#define TTHEN 9 /* then : Keyword */
```

token-list.h の 20 行目に定義があります。

4.4.1.49 TTRUE

```
#define TTRUE 25 /* true : Keyword */
```

token-list.h の 36 行目に定義があります。

4.4.1.50 TVAR

```
#define TVAR 3 /* var : Keyword */
```

token-list.h の 14 行目に定義があります。

4.4.1.51 TWHILE

```
#define TWHILE 14 /* while : Keyword */
```

token-list.h の 25 行目に定義があります。

4.4.1.52 TWRITE

```
#define TWRITE 48 /* write : Keyword */
```

token-list.h の 59 行目に定義があります。

4.4.1.53 TWRITELN

```
#define TWRITELN 24 /* writeln : Keyword */
```

token-list.h の 35 行目に定義があります。

4.4.2 関数詳解

4.4.2.1 end_scan()

```
int endscan (
    void )
```

scan.c の 89 行目に定義があります。

```
89      {
90      if (fclose(fp) == EOF) {
91          return -1;
92      }
93      return 0;
94 }
```

被呼び出し関係図:



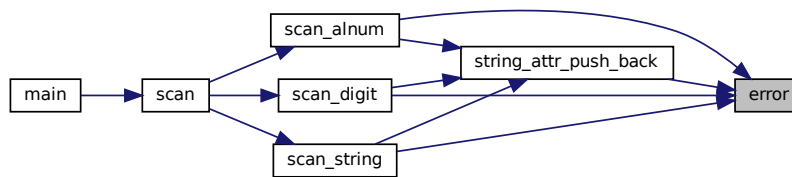
4.4.2.2 error()

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

token-list.c の 87 行目に定義があります。

```
87     {
88         fprintf(stderr, "\n ERROR: %s\n", mes);
89         /* end_scan(); */
90     }
```

被呼び出し関係図:



4.4.2.3 get_linenum()

```
int get_linenum (
    void )
```

scan.c の 81 行目に定義があります。

```
81     {
82         return token_linenum;
83     }
```

4.4.2.4 init_scan()

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

scan.c の 26 行目に定義があります。

```
26     {
27         if ((fp = fopen(filename, "r")) == NULL) {
28             return -1;
29         }
30
31         look_ahead();
32         look_ahead();
33
34         return 0;
35     }
```

呼び出し関係図:



被呼び出し関係図:



4.4.2.5 scan()

```
int scan (
    void )
```

scan.c の 37 行目に定義があります。

```

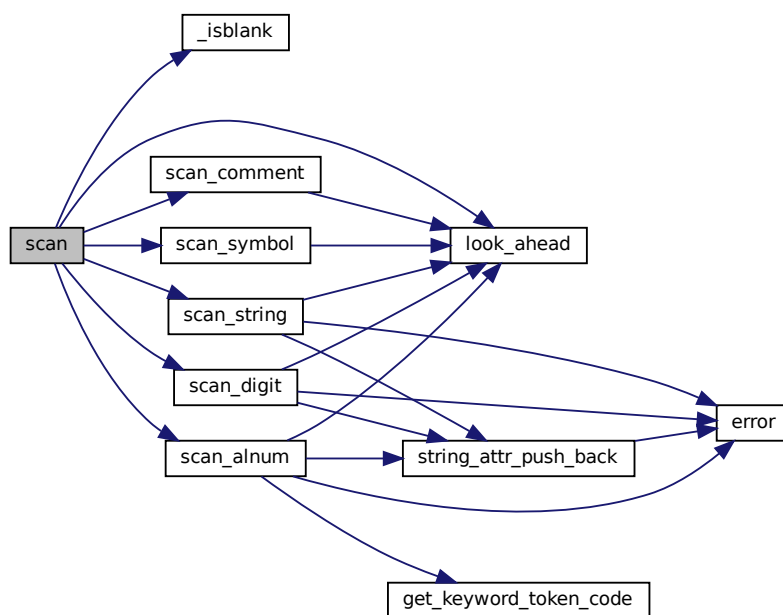
37     {
38         int token_code = -1;
39         while (1) {
40             if (current_char == EOF) { /* End Of File*/
41                 return -1;
42             } else if (current_char == '\r' || current_char == '\n') { /* End of Line */
43                 if (current_char == '\r') {
44                     if (next_char == '\n') {
45                         look_ahead();
46                     }
47                     look_ahead();
48                     linenum++;
49                 } else {
50                     if (next_char == '\r') {
51                         look_ahead();
52                     }
53                     look_ahead();
54                     linenum++;
55                 }
56             } else if (!isblank(current_char)) { /* Separator (Space or Tab) */
57                 look_ahead();
58             } else if (!isprint(current_char)) { /* Not Graphic Character(0x20~0x7e) */
59                 return -1;
60             } else if (isalpha(current_char)) { /* Name or Keyword */
61                 token_code = scan_alnum();
62                 break;
63             } else if (isdigit(current_char)) { /* Digit */
64                 token_code = scan_digit();
65                 break;
66             } else if (current_char == '\'') { /* String */
67                 token_code = scan_string();
68                 break;
69             } else if ((current_char == '/' && next_char == '*') || current_char == '{') { /* Comment */
```

```

70         if (scan_comment() == -1) {
71             break;
72         }
73     } else { /* Symbol */
74         token_code = scan_symbol();
75         break;
76     }
77 }
78 return token_code;
79 }

```

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]

scan.c の 8 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]
```

scan.c の 24 行目に定義があります。

4.4.3.4 string_attr

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

scan.c の 9 行目に定義があります。

Index

- `_isblank`
 - `scan.c`, 11
- `count`
 - `ID`, 5
- `current_char`
 - `scan.c`, 22
- `end_scan`
 - `scan.c`, 12
 - `token-list.h`, 38
- `error`
 - `token-list.c`, 24
 - `token-list.h`, 38
- `fp`
 - `scan.c`, 22
 - `token-list.h`, 41
- `get_keyword_token_code`
 - `scan.c`, 12
- `get_linenum`
 - `scan.c`, 13
 - `token-list.h`, 39
- `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`, 13
 - `token-list.h`, 39
- `KEY`, 6
 - `keytoken`, 6
 - `keyword`, 6
- `key`
 - `token-list.c`, 26
 - `token-list.h`, 41
- `keytoken`
 - `KEY`, 6
- `keyword`
 - `KEY`, 6
- `KEYWORDSIZE`
 - `token-list.h`, 29
- `linenum`
 - `scan.c`, 23
- `look_ahead`
 - `scan.c`, 14
- `main`
 - `token-list.c`, 25
- `MAX_NUM_ATTR`
 - `token-list.h`, 29
- `MAXSTRSIZE`
 - `token-list.h`, 30
- `name`
 - `ID`, 5
- `next_char`
 - `scan.c`, 23
- `nextp`
 - `ID`, 6
- `num_attr`
 - `scan.c`, 23
 - `token-list.h`, 42
- `NUMOFTOKEN`
 - `token-list.h`, 30
- `numtoken`
 - `token-list.c`, 26
- `print_idtab`
 - `id-list.c`, 9
- `release_idtab`
 - `id-list.c`, 9
- `scan`
 - `scan.c`, 15
 - `token-list.h`, 40
- `scan.c`, 10
 - `_isblank`, 11
 - `current_char`, 22
 - `end_scan`, 12
 - `fp`, 22
 - `get_keyword_token_code`, 12
 - `get_linenum`, 13
 - `init_scan`, 13

- linenum, 23
- look_ahead, 14
- next_char, 23
- num_attr, 23
- scan, 15
- scan_alnum, 16
- scan_comment, 17
- scan_digit, 18
- scan_string, 19
- scan_symbol, 20
- set_token_linenum, 21
- string_attr, 23
- string_attr_push_back, 21
- token_linenum, 23
- scan_alnum
 - scan.c, 16
- scan_comment
 - scan.c, 17
- scan_digit
 - scan.c, 18
- scan_string
 - scan.c, 19
- scan_symbol
 - scan.c, 20
- search_idtab
 - id-list.c, 9
- set_token_linenum
 - scan.c, 21
- string_attr
 - scan.c, 23
 - token-list.h, 42
- string_attr_push_back
 - scan.c, 21
- TAND
 - token-list.h, 30
- TARRAY
 - token-list.h, 30
- TASSIGN
 - token-list.h, 30
- TBEGIN
 - token-list.h, 30
- TBOOLEAN
 - token-list.h, 31
- TBREAK
 - token-list.h, 31
- TCALL
 - token-list.h, 31
- TCHAR
 - token-list.h, 31
- TCOLON
 - token-list.h, 31
- TCOMMA
 - token-list.h, 31
- TDIV
 - token-list.h, 32
- TDO
 - token-list.h, 32
- TDOT
 - token-list.h, 32
- TELSE
 - token-list.h, 32
- TEND
 - token-list.h, 32
- TEQUAL
 - token-list.h, 32
- TFALSE
 - token-list.h, 33
- TGR
 - token-list.h, 33
- TGREQ
 - token-list.h, 33
- TIF
 - token-list.h, 33
- TINTEGER
 - token-list.h, 33
- TLE
 - token-list.h, 33
- TLEEQ
 - token-list.h, 34
- TLPAREN
 - token-list.h, 34
- TLSQPAREN
 - token-list.h, 34
- TMINUS
 - token-list.h, 34
- TNAME
 - token-list.h, 34
- TNOT
 - token-list.h, 34
- TNOTEQ
 - token-list.h, 35
- TNUMBER
 - token-list.h, 35
- TOF
 - token-list.h, 35
- token-list.c, 24
 - error, 24
 - key, 26
 - main, 25
 - numtoken, 26
 - tokenstr, 27
- token-list.h, 27
 - end_scan, 38
 - error, 38
 - fp, 41
 - get_linenum, 39
 - init_scan, 39
 - key, 41
 - KEYWORDSIZE, 29
 - MAX_NUM_ATTR, 29
 - MAXSTRSIZE, 30
 - num_attr, 42
 - NUMOFTOKEN, 30
 - scan, 40
 - string_attr, 42
 - TAND, 30

- TARRAY, 30
 - TASSIGN, 30
 - TBEGIN, 30
 - TBOOLEAN, 31
 - TBREAK, 31
 - TCALL, 31
 - TCHAR, 31
 - TCOLON, 31
 - TCOMMA, 31
 - TDIV, 32
 - TDO, 32
 - TDOT, 32
 - TELSE, 32
 - TEND, 32
 - TEQUAL, 32
 - TFALSE, 33
 - TGR, 33
 - TGREQ, 33
 - TIF, 33
 - TINTEGER, 33
 - TLE, 33
 - TLEEQ, 34
 - TLPAREN, 34
 - TLSPAREN, 34
 - TMINUS, 34
 - TNAME, 34
 - TNOT, 34
 - TNOTEQ, 35
 - TNUMBER, 35
 - TOF, 35
 - TOR, 35
 - TPLUS, 35
 - TPROCEDURE, 35
 - TPROGRAM, 36
 - token-list.h, 36
 - TREAD, 36
 - token-list.h, 36
 - TREADLN, 36
 - token-list.h, 36
 - TRETURN, 36
 - token-list.h, 36
 - TRPAREN, 36
 - token-list.h, 36
 - TRSQPAREN, 36
 - token-list.h, 36
 - TSEMI, 37
 - token-list.h, 37
 - TSTAR, 37
 - token-list.h, 37
 - TSTRING, 37
 - token-list.h, 37
 - TTHEN, 37
 - token-list.h, 37
 - TTRUE, 37
 - token-list.h, 37
 - TVAR, 37
 - token-list.h, 37
 - TWHILE, 38
 - token-list.h, 38
 - TWRITE, 38
 - token-list.h, 38
 - TWRITELN, 38
 - token-list.h, 38
- token_linenum
- scan.c, 23
- tokenstr
- token-list.c, 27
- TOR
- token-list.h, 35
- TPLUS
- token-list.h, 35
- TPROCEDURE
- token-list.h, 35