# tetris

0.0.1

Generated by Doxygen 1.9.1

# Chapter 1

# c_game_project

ubuntu 22.04 only using C language game project with ncurses library

## 1.1 2024-03-12

- project start
- Doxyfile
- README.md
- .gitignore
- .vscode/
  - tasks.json
  - launch.json
  - settings.json
  - c_cpp_properties.json
- :
  - https://blog.naver.com/hardinge/220515010763
-
  - : tetrisV01.c
  - : display_menu.c, display_tetris.c
  - : result.c
  - : update.c
  - : getch.c
- cmakefiles
  -
    - file(GLOB SRC_FILES "∗.c")
  -
    - install(TARGETS tetrisV01 DESTINATION /usr/local/bin)
  -
    - install(DIRECTORY ${CMAKE_SOURCE_DIR}/include/ DESTINATION /usr/local/include)
    - install(DIRECTORY ${CMAKE_SOURCE_DIR}/src/ DESTINATION /usr/local/src FILES_↩ MATCHING PATTERN "∗.c")
  -

# Chapter 2

# Class Index

## 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

# Chapter 3

# File Index

## 3.1 File List

Here is a list of all files with brief descriptions:

# Chapter 4

# Class Documentation

## 4.1 result Struct Reference

`#include <result.h>`
Collaboration diagram for result:


### Public Attributes

- char name [30]
- long point
- int year
- int month
- int day
- int hour
- int min
- int rank

### 4.1.1 Detailed Description

,
Definition at line 14 of file result.h.

### 4.1.2 Member Data Documentation

#### 4.1.2.1 day

`int result::day`
Definition at line 20 of file result.h.

#### 4.1.2.2 hour

`int result::hour`
Definition at line 21 of file result.h.

#### 4.1.2.3 min

`int result::min`
Definition at line 22 of file result.h.

#### 4.1.2.4 month

`int result::month`
Definition at line 19 of file result.h.

#### 4.1.2.5 name

`char result::name[30]`
Definition at line 16 of file result.h.

#### 4.1.2.6 point

`long result::point`
Definition at line 17 of file result.h.

#### 4.1.2.7 rank

`int result::rank`
Definition at line 23 of file result.h.

#### 4.1.2.8 year

`int result::year`
Definition at line 18 of file result.h.
The documentation for this struct was generated from the following file:

- include/result.h

# Chapter 5

# File Documentation

## 5.1 build/CMakeFiles/3.22.1/CompilerIdC/CMakeCCompilerId.c File Reference

### Macros

- #define __has_include(x) 0
- #define COMPILER_ID ""
- #define STRINGIFY_HELPER(X) #X
- #define STRINGIFY(X) STRINGIFY_HELPER(X)
- #define PLATFORM_ID
- #define ARCHITECTURE_ID
- #define DEC(n)
- #define HEX(n)
- #define C_VERSION

### Functions

- int main (int argc, char *argv[])

### Variables

- char const * info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
- char const * info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
- char const * info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
- const char * info_language_standard_default
- const char * info_language_extensions_default

### 5.1.1 Macro Definition Documentation

#### 5.1.1.1 __has_include

```
#define __has_include(
              x ) 0
```

Definition at line 17 of file CMakeCCompilerId.c.

#### 5.1.1.2 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

Definition at line 652 of file CMakeCCompilerId.c.

#### 5.1.1.3 C_VERSION

```
#define C_VERSION
```

Definition at line 741 of file CMakeCCompilerId.c.

#### 5.1.1.4 COMPILER_ID

```
#define COMPILER_ID ""
```

Definition at line 396 of file CMakeCCompilerId.c.

#### 5.1.1.5 DEC

```
#define DEC(
              n )
```

**Value:**

```
  ('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

Definition at line 656 of file CMakeCCompilerId.c.

#### 5.1.1.6 HEX

```
#define HEX(
              n )
```

**Value:**

```
  ('0' + ((n)>>28 & 0xF)), \
  ('0' + ((n)>>24 & 0xF)), \
  ('0' + ((n)>>20 & 0xF)), \
  ('0' + ((n)>>16 & 0xF)), \
  ('0' + ((n)>>12 & 0xF)), \
  ('0' + ((n)>>8  & 0xF)), \
  ('0' + ((n)>>4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

Definition at line 667 of file CMakeCCompilerId.c.

#### 5.1.1.7 PLATFORM_ID

```
#define PLATFORM_ID
```

Definition at line 524 of file CMakeCCompilerId.c.

#### 5.1.1.8 STRINGIFY

```
#define STRINGIFY(
              X ) STRINGIFY_HELPER(X)
```

Definition at line 417 of file CMakeCCompilerId.c.

#### 5.1.1.9 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(
              X ) #X
```

Definition at line 416 of file CMakeCCompilerId.c.

## 5.1.2 Function Documentation

### 5.1.2.1 main()

```
int main (
            int argc,
            char * argv[] )
```

Definition at line 776 of file CMakeCCompilerId.c.

```
778 {
779   int require = 0;
780   require += info_compiler[argc];
781   require += info_platform[argc];
782   require += info_arch[argc];
783 #ifdef COMPILER_VERSION_MAJOR
784   require += info_version[argc];
785 #endif
786 #ifdef COMPILER_VERSION_INTERNAL
787   require += info_version_internal[argc];
788 #endif
789 #ifdef SIMULATE_ID
790   require += info_simulate[argc];
791 #endif
792 #ifdef SIMULATE_VERSION_MAJOR
793   require += info_simulate_version[argc];
794 #endif
795 #if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
796   require += info_cray[argc];
797 #endif
798   require += info_language_standard_default[argc];
799   require += info_language_extensions_default[argc];
800   (void)argv;
801   return require;
802 }
```

## 5.1.3 Variable Documentation

### 5.1.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

Definition at line 733 of file CMakeCCompilerId.c.

### 5.1.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

Definition at line 403 of file CMakeCCompilerId.c.

### 5.1.3.3 info_language_extensions_default

```
const char* info_language_extensions_default
```

**Initial value:**

```
= "INFO" ":" "extensions_default["
  "OFF"
"]"
```

Definition at line 757 of file CMakeCCompilerId.c.

### 5.1.3.4 info_language_standard_default

```
const char* info_language_standard_default
```

**Initial value:**

```
=
  "INFO" ":" "standard_default[" C_VERSION "]"
```

Definition at line 754 of file CMakeCCompilerId.c.

#### 5.1.3.5 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```
Definition at line 732 of file CMakeCCompilerId.c.

## 5.2 build/CMakeFiles/3.22.1/CompilerIdCXX/CMakeCXXCompilerId.cpp File Reference

### Macros

- #define __has_include(x) 0
- #define COMPILER_ID ""
- #define STRINGIFY_HELPER(X) #X
- #define STRINGIFY(X) STRINGIFY_HELPER(X)
- #define PLATFORM_ID
- #define ARCHITECTURE_ID
- #define DEC(n)
- #define HEX(n)
- #define CXX_STD __cplusplus

### Functions

- int main (int argc, char *argv[])

### Variables

- char const * info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
- char const * info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
- char const * info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
- const char * info_language_standard_default
- const char * info_language_extensions_default

### 5.2.1 Macro Definition Documentation

#### 5.2.1.1 __has_include

```
#define __has_include(
              x ) 0
```
Definition at line 11 of file CMakeCXXCompilerId.cpp.

#### 5.2.1.2 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```
Definition at line 637 of file CMakeCXXCompilerId.cpp.

#### 5.2.1.3 COMPILER_ID

```
#define COMPILER_ID ""
```
Definition at line 381 of file CMakeCXXCompilerId.cpp.

#### 5.2.1.4 CXX_STD

```
#define CXX_STD __cplusplus
```

Definition at line 735 of file CMakeCXXCompilerId.cpp.

#### 5.2.1.5 DEC

```
#define DEC(
              n )
```

**Value:**

```
  ('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

Definition at line 641 of file CMakeCXXCompilerId.cpp.

#### 5.2.1.6 HEX

```
#define HEX(
              n )
```

**Value:**

```
  ('0' + ((n)>>28 & 0xF)), \
  ('0' + ((n)>>24 & 0xF)), \
  ('0' + ((n)>>20 & 0xF)), \
  ('0' + ((n)>>16 & 0xF)), \
  ('0' + ((n)>>12 & 0xF)), \
  ('0' + ((n)>>8  & 0xF)), \
  ('0' + ((n)>>4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

Definition at line 652 of file CMakeCXXCompilerId.cpp.

#### 5.2.1.7 PLATFORM_ID

```
#define PLATFORM_ID
```

Definition at line 509 of file CMakeCXXCompilerId.cpp.

#### 5.2.1.8 STRINGIFY

```
#define STRINGIFY(
              X ) STRINGIFY_HELPER(X)
```

Definition at line 402 of file CMakeCXXCompilerId.cpp.

#### 5.2.1.9 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(
              X ) #X
```

Definition at line 401 of file CMakeCXXCompilerId.cpp.

### 5.2.2 Function Documentation

#### 5.2.2.1 main()

```
int main (
             int argc,
```

```
            char * argv[] )
```

Definition at line 767 of file CMakeCXXCompilerId.cpp.

```
768 {
769   int require = 0;
770   require += info_compiler[argc];
771   require += info_platform[argc];
772 #ifdef COMPILER_VERSION_MAJOR
773   require += info_version[argc];
774 #endif
775 #ifdef COMPILER_VERSION_INTERNAL
776   require += info_version_internal[argc];
777 #endif
778 #ifdef SIMULATE_ID
779   require += info_simulate[argc];
780 #endif
781 #ifdef SIMULATE_VERSION_MAJOR
782   require += info_simulate_version[argc];
783 #endif
784 #if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
785   require += info_cray[argc];
786 #endif
787   require += info_language_standard_default[argc];
788   require += info_language_extensions_default[argc];
789   (void)argv;
790   return require;
791 }
```

### 5.2.3 Variable Documentation

#### 5.2.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

Definition at line 718 of file CMakeCXXCompilerId.cpp.

#### 5.2.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

Definition at line 388 of file CMakeCXXCompilerId.cpp.

#### 5.2.3.3 info_language_extensions_default

```
const char* info_language_extensions_default
```

**Initial value:**

```
= "INFO" ":" "extensions_default["
  "OFF"
"]"
```

Definition at line 754 of file CMakeCXXCompilerId.cpp.

#### 5.2.3.4 info_language_standard_default

```
const char* info_language_standard_default
```

**Initial value:**

```
= "INFO" ":" "standard_default["
  "98"
"]"
```

Definition at line 738 of file CMakeCXXCompilerId.cpp.

#### 5.2.3.5 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

Definition at line 717 of file CMakeCXXCompilerId.cpp.

## 5.3 build/CMakeFiles/tetrisV01.dir/src/display_menu.c.o.d File Reference

## 5.4 build/CMakeFiles/tetrisV01.dir/src/display_tetris.c.o.d File Reference

## 5.5 build/CMakeFiles/tetrisV01.dir/src/getch.c.o.d File Reference

## 5.6 build/CMakeFiles/tetrisV01.dir/src/result.c.o.d File Reference

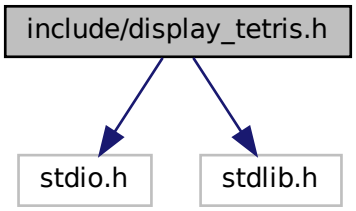
## 5.7 build/CMakeFiles/tetrisV01.dir/src/tetrisV01.c.o.d File Reference

## 5.8 build/CMakeFiles/tetrisV01.dir/src/update.c.o.d File Reference

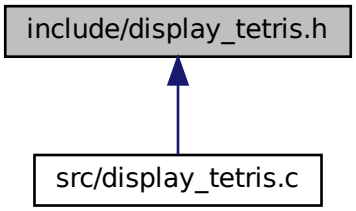
## 5.9 include/display_tetris.h File Reference

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

Include dependency graph for display_tetris.h:




This graph shows which files directly or indirectly include this file:




### Enumerations

- enum BlockType {
  I_BLOCK , T_BLOCK , S_BLOCK , Z_BLOCK ,
  L_BLOCK , J_BLOCK , O_BLOCK , I_BLOCK ,
  T_BLOCK , S_BLOCK , Z_BLOCK , L_BLOCK ,

J_BLOCK , O_BLOCK , I_BLOCK , T_BLOCK , S_BLOCK , Z_BLOCK , L_BLOCK , J_BLOCK , O_BLOCK }

## Functions

- int display_tetris_table (int *)

## Variables

- char i_block [4][4][4]
- char t_block [4][4][4]
- char s_block [4][4][4]
- char z_block [4][4][4]
- char l_block [4][4][4]
- char j_block [4][4][4]
- char o_block [4][4][4]
- int next_block_number
- long point
- int best_point
- char tetris_table [21][10]

### 5.9.1 Enumeration Type Documentation

#### 5.9.1.1 BlockType

```
enum BlockType
```

**Enumerator**

| | |
|---|---|
| I_BLOCK | |
| T_BLOCK | |
| S_BLOCK | |
| Z_BLOCK | |
| L_BLOCK | |
| J_BLOCK | |
| O_BLOCK | |
| I_BLOCK | |
| T_BLOCK | |
| S_BLOCK | |
| Z_BLOCK | |
| L_BLOCK | |
| J_BLOCK | |
| O_BLOCK | |
| I_BLOCK | |
| T_BLOCK | |
| S_BLOCK | |
| Z_BLOCK | |
| L_BLOCK | |
| J_BLOCK | |
| O_BLOCK | |

Definition at line 7 of file display_tetris.h.

```
8 {
9     I_BLOCK,
10     T_BLOCK,
11     S_BLOCK,
12     Z_BLOCK,
13     L_BLOCK,
14     J_BLOCK,
15     O_BLOCK
16 };
```

## 5.9.2 Function Documentation

### 5.9.2.1 display_tetris_table()

```
int display_tetris_table (
            int * countrange )
```

Definition at line 4 of file display_tetris.c.

```
5 {
6     int i, j;
7     char(*block_pointer)[4][4][4] = NULL;
8
9     switch (next_block_number)
10     {
11     case I_BLOCK:
12         block_pointer = &i_block;
13         break;
14     case T_BLOCK:
15         block_pointer = &t_block;
16         break;
17     case S_BLOCK:
18         block_pointer = &s_block;
19         break;
20     case Z_BLOCK:
21         block_pointer = &z_block;
22         break;
23     case L_BLOCK:
24         block_pointer = &l_block;
25         break;
26     case J_BLOCK:
27         block_pointer = &j_block;
28         break;
29     case O_BLOCK:
30         block_pointer = &o_block;
31         break;
32     }
33
34     system("clear");
35     printf("\n Score: %ld | Speed: %d | hihgest score: %d", point, *countrange, best_point);
36
37     printf("\n\n Next Block\n");
38
39     for (i = 0; i < 4; i++)
40     {
41         printf("\n ");
42         for (j = 0; j < 4; j++)
43         {
44             if ((*block_pointer)[0][i][j] == 1)
45                 printf("");
46             else if ((*block_pointer)[0][i][j] == 0)
47                 printf(" ");
48         }
49     }
50
51     for (i = 2; i < 21; i++)
52     {
53         printf("\t");
54         for (j = 0; j < 10; j++)
55         {
56             if (j == 0 || j == 9 || (i == 20 && (j > 1 || j < 9)))
57             {
58                 printf("");
59             }
60             else if (tetris_table[i][j] == 1)
61                 printf("");
62             else if (tetris_table[i][j] == 0)
63                 printf(" ");
64         }
65         printf("\n");
66     }
67     printf("\n GAME STOP : P");
```

```
68     return 0;
69 }
```

Here is the caller graph for this function:


### 5.9.3 Variable Documentation

#### 5.9.3.1 best_point

```
int best_point  [extern]
```

Definition at line 47 of file tetrisV01.h.

#### 5.9.3.2 i_block

```
char i_block[4][4][4]  [extern]
```

Definition at line 40 of file update.h.

#### 5.9.3.3 j_block

```
char j_block[4][4][4]  [extern]
```

Definition at line 135 of file update.h.

#### 5.9.3.4 l_block

```
char l_block[4][4][4]  [extern]
```

Definition at line 116 of file update.h.

#### 5.9.3.5 next_block_number

```
int next_block_number  [extern]
```

Definition at line 175 of file update.h.

#### 5.9.3.6 o_block

```
char o_block[4][4][4]  [extern]
```

Definition at line 154 of file update.h.

#### 5.9.3.7 point

```
long point  [extern]
```

Definition at line 48 of file tetrisV01.h.

#### 5.9.3.8 s_block

```
char s_block[4][4][4]  [extern]
```
Definition at line 78 of file update.h.

#### 5.9.3.9 t_block

```
char t_block[4][4][4]  [extern]
```
Definition at line 59 of file update.h.

#### 5.9.3.10 tetris_table

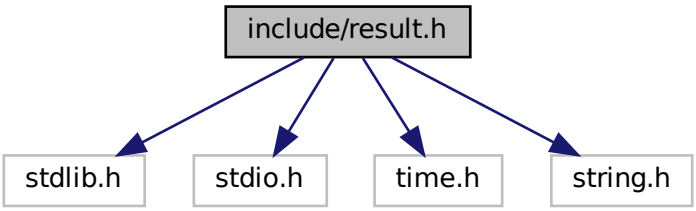
```
char tetris_table[21][10]  [extern]
```
Definition at line 43 of file tetrisV01.h.

#### 5.9.3.11 z_block

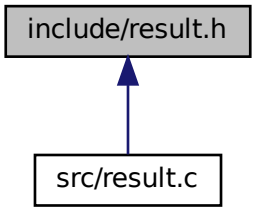
```
char z_block[4][4][4]  [extern]
```
Definition at line 97 of file update.h.

## 5.10 include/result.h File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <string.h>
```
Include dependency graph for result.h:



This graph shows which files directly or indirectly include this file:




## Classes

- struct result

## Functions

- int print_result (void)
- int search_result (void)
- int save_result (long, int)
- int getch (void)

## Variables

- static struct result temp_result

### 5.10.1 Function Documentation

#### 5.10.1.1 getch()

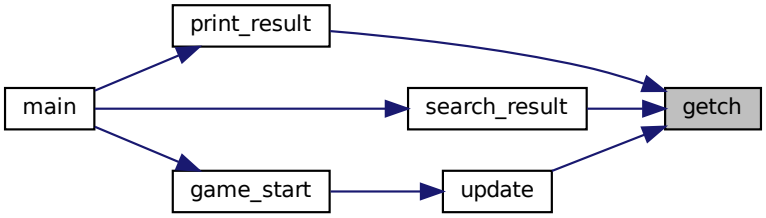
```
int getch (
            void  )
```

Definition at line 14 of file getch.c.

```
15 {
16              char   ch;
17              int   error;
18              static struct termios Otty, Ntty;
19
20              fflush(stdout);
21              tcgetattr(0, &Otty);
22              Ntty = Otty;
23              Ntty.c_iflag  =  0;
24              Ntty.c_oflag  =  0;
25              Ntty.c_lflag &= ~ICANON;
26 #if 1
27              Ntty.c_lflag &= ~ECHO;
28 #else
29              Ntty.c_lflag |=  ECHO;
30 #endif
31              Ntty.c_cc[VMIN]  = CCHAR;
32              Ntty.c_cc[VTIME] = CTIME;
33
34 #if 1
35 #define FLAG TCSAFLUSH
36 #else
37 #define FLAG TCSANOW
38 #endif
39
40              if (0 == (error = tcsetattr(0, FLAG, &Ntty)))
```

```
41              {
42                      error  = read(0, &ch, 1 );
43                      error += tcsetattr(0, FLAG, &Otty);
44              }
45
46              return (error == 1 ? (int) ch : -1 );
47 }
```

Here is the caller graph for this function:



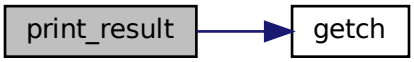

#### 5.10.1.2 print_result()

```
int print_result (
              void  )
```

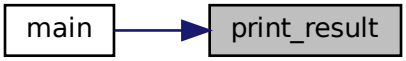
Definition at line 98 of file result.c.

```
99 {
100     //
101     char *home_dir = getenv("HOME");
102     char result_file[256];
103     sprintf(result_file, "%s/result", home_dir);
104
105     FILE *fp = NULL;
106     char ch = 1;
107
108     fp = fopen(result_file, "rb");
109
110     if (fp == NULL)
111         return 0;
112
113     system("clear");
114
115     printf("\n\t\t\t\tText Tetris");
116     printf("\n\t\t\t\t Game Stats\n\n");
117     printf("\n\t\tName\t\tScore\t   Date\t\t Time");
118
119     while (1)
120     {
121         fread(&temp_result, sizeof(struct result), 1, fp);
122         if (!feof(fp))
123         {
124             printf("\n\t=============================================================");
125             printf("\n\t\t%s\n\t\t\t\t%ld\t %d.%d.%d.   |   %d:%d\n", temp_result.name, temp_result.point,
    temp_result.year, temp_result.month, temp_result.day, temp_result.hour, temp_result.min);
126         }
127         else
128         {
129             break;
130         }
131     }
132
133     fclose(fp);
134
135     printf("\n\n\tBack to the game menu : M");
136     while (1)
137     {
138         ch = getch();
139         if (ch == 77 || ch == 109)
140             break;
141     }
142     return 0;
143 }
```

Here is the call graph for this function:




Here is the caller graph for this function:




#### 5.10.1.3 save_result()

```
int save_result (
            long point,
            int best_point )
```

Definition at line 4 of file result.c.

```
5 {
6     //
7     char *home_dir = getenv("HOME");
8     char result_file[256];
9     sprintf(result_file, "%s/result", home_dir);
10
11     FILE *fp = NULL;
12     time_t ptime;
13     struct tm *t;
14     //
15     printf("\n\n Final score : %ld ", point);
16     printf("\n\n Please enter your name : ");
17     scanf("%s%*c", temp_result.name);
18     temp_result.point = point;
19
20     if (temp_result.point >= best_point)
21         best_point = temp_result.point;
22
23     ptime = time(NULL);    //
24     t = localtime(&ptime); //
25
26     temp_result.year = t->tm_year + 1900;
27     temp_result.month = t->tm_mon + 1;
28     temp_result.day = t->tm_mday;
29     temp_result.hour = t->tm_hour;
30     temp_result.min = t->tm_min;
31
32     fp = fopen(result_file, "ab");
33     fseek(fp, 1, SEEK_END);
34     fwrite(&temp_result, sizeof(struct result), 1, fp);
35     fclose(fp);
36     return 0;
37 }
```

Here is the caller graph for this function:

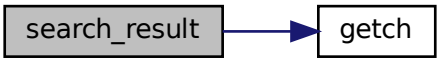

#### 5.10.1.4 search_result()

```
int search_result (
            void  )
```

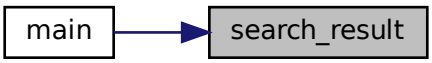
Definition at line 39 of file result.c.

```
40 {
41     //
42     char *home_dir = getenv("HOME");
43     char result_file[256];
44     sprintf(result_file, "%s/result", home_dir);
45 
46     FILE *fp = NULL;
47     char name[30];
48     char ch;
49     int find = 0;
50 
51     fp = fopen(result_file, "rb");
52 
53     if (fp == NULL)
54         return 0;
55 
56     system("clear");
57 
58     printf("\n\n\t\tEnter the name your to search.  : ");
59     scanf("%s%*c", name);
60 
61     printf("\n\t\t\t\tText Tetris");
62     printf("\n\t\t\t\t Game Stats\n\n");
63     printf("\n\t\tName\t\tScore\t   Date\t\t Time");
64 
65     while (1)
66     {
67         fread(&temp_result, sizeof(struct result), 1, fp);
68         if (!feof(fp))
69         {
70             if (!strcmp(temp_result.name, name))
71             {
72                 find = 1;
73                 printf("\n\t============================================================");
74                 printf("\n\t\t%s\n\t\t\t\t%ld\t%d.%d.%d.  |  %d:%d\n", temp_result.name,
    temp_result.point, temp_result.year, temp_result.month, temp_result.day, temp_result.hour,
    temp_result.min);
75             }
76         }
77         else
78         {
79             break;
80         }
81     }
82 
83     if (find == 0)
84         printf("\n\n\n\t\tThis name is not found.");
85 
86     printf("\n\n\n\t\tBack to the game menu : M");
87     while (1)
88     {
89         ch = getch();
90         if (ch == 77 || ch == 109)
91             break;
92     }
93 
94     return 0;
95 }
```

Here is the call graph for this function:




Here is the caller graph for this function:




### 5.10.2 Variable Documentation

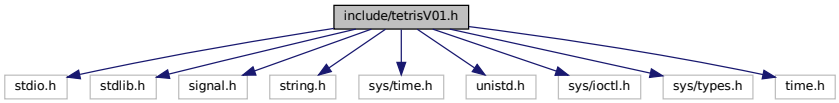
#### 5.10.2.1 temp_result

```
struct result temp_result  [static]
```

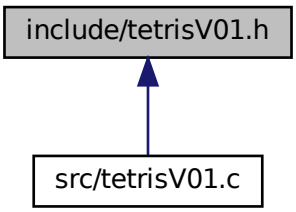
## 5.11 include/tetrisV01.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <string.h>
#include <sys/time.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <time.h>
```

Include dependency graph for tetrisV01.h:



This graph shows which files directly or indirectly include this file:




## Macros

- #define GAME_START 0
- #define GAME_END 1

## Enumerations

- enum Direction {
  LEFT , RIGHT , DOWN , ROTATE ,
  LEFT , RIGHT , DOWN , ROTATE }
- enum BlockType {
  I_BLOCK , T_BLOCK , S_BLOCK , Z_BLOCK ,
  L_BLOCK , J_BLOCK , O_BLOCK , I_BLOCK ,
  T_BLOCK , S_BLOCK , Z_BLOCK , L_BLOCK ,
  J_BLOCK , O_BLOCK , I_BLOCK , T_BLOCK ,
  S_BLOCK , Z_BLOCK , L_BLOCK , J_BLOCK ,
  O_BLOCK }

## Functions

- int init_tetris_table (void)
- int game_start (void)
- int display_menu (void)
- int update (int)
- int print_result (void)
- int search_result (void)
- int save_result (long, int)
- int getch (void)

## Variables

- char tetris_table [21][10]
- int x = 3
- int y = 0
- int game = GAME_END
- int best_point = 0
- long point = 0

### 5.11.1 Macro Definition Documentation

#### 5.11.1.1 GAME_END

```
#define GAME_END 1
```

Definition at line 36 of file tetrisV01.h.

#### 5.11.1.2 GAME_START

```
#define GAME_START 0
```

Definition at line 35 of file tetrisV01.h.

### 5.11.2 Enumeration Type Documentation

#### 5.11.2.1 BlockType

```
enum BlockType
```

**Enumerator**

| | |
|---|---|
| I_BLOCK | |
| T_BLOCK | |
| S_BLOCK | |
| Z_BLOCK | |
| L_BLOCK | |
| J_BLOCK | |
| O_BLOCK | |
| I_BLOCK | |
| T_BLOCK | |
| S_BLOCK | |
| Z_BLOCK | |
| L_BLOCK | |
| J_BLOCK | |
| O_BLOCK | |
| I_BLOCK | |
| T_BLOCK | |
| S_BLOCK | |
| Z_BLOCK | |
| L_BLOCK | |
| J_BLOCK | |
| O_BLOCK | |

Definition at line 23 of file tetrisV01.h.

```
24 {
25     I_BLOCK,
26     T_BLOCK,
27     S_BLOCK,
28     Z_BLOCK,
29     L_BLOCK,
30     J_BLOCK,
31     O_BLOCK
32 };
```

#### 5.11.2.2 Direction

```
enum Direction
```

**Enumerator**

| | |
|---|---|
| LEFT | |
| RIGHT | |
| DOWN | |
| ROTATE | |
| LEFT | |
| RIGHT | |
| DOWN | |
| ROTATE | |

Definition at line 15 of file tetrisV01.h.

```
16 {
17     LEFT,
18     RIGHT,
19     DOWN,
20     ROTATE
21 };
```

## 5.11.3 Function Documentation

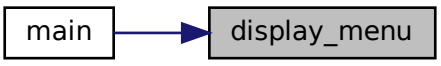
### 5.11.3.1 display_menu()

```
int display_menu (
              void  )
```

Definition at line 5 of file display_menu.c.

```
6 {
7     int menu = 0;
8
9     while(1)
10     {
11         system("clear");
12         printf("\n\n\t\t\t\tText Tetris");
13         printf("\n\t\t\t============================");
14         printf("\n\t\t\t\tGAME MENU\t\n");
15         printf("\n\t\t\t============================");
16         printf("\n\t\t\t   1) Game Start");
17         printf("\n\t\t\t   2) Search history");
18         printf("\n\t\t\t   3) Record Output");
19         printf("\n\t\t\t   4) QUIT");
20         printf("\n\t\t\t============================");
21         printf("\n\t\t\t\t\t SELECT : ");
22         scanf("%d",&menu);
23         if(menu < 1 || menu > 4)
24         {
25             continue;
26         }
27         else
28         {
29             return menu;
30         }
31     }
32     return 0;
33 }
```

Here is the caller graph for this function:



#### 5.11.3.2 game_start()

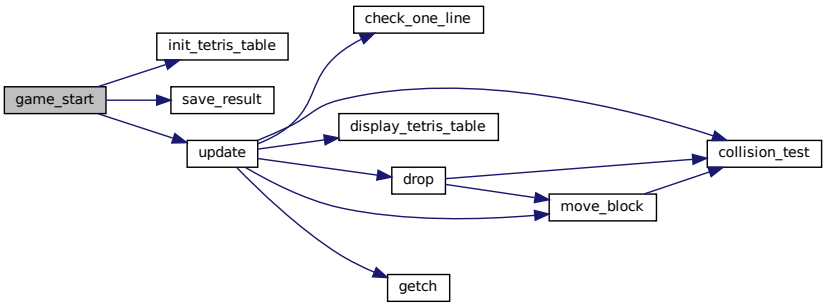
```
int game_start (
            void  )
```

. game . update() signal .

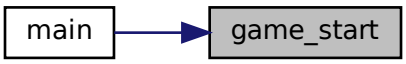
Definition at line 38 of file tetrisV01.c.

```
39 {
40
41     if (game == GAME_START)
42     {
43         init_tetris_table();
44
45         //
46         static struct itimerval timer;
47         //  type    . signal(SIGALRM, (void (*)(int))update);
48         signal(SIGVTALRM, (void (*)(int))update);
49         timer.it_value.tv_sec = 0;
50         timer.it_value.tv_usec = 1;
51         timer.it_interval.tv_sec = 0;
52         timer.it_interval.tv_usec = 1;
53         setitimer(ITIMER_VIRTUAL, &timer, NULL);
54
55         while (1)
56         {
57             if (game == GAME_END)
58             {
59                 signal(SIGALRM, SIG_IGN);
60                 save_result(point, best_point);
61
62                 x = 3, y = 0;
63                 point = 0;
64
65                 return 1;
66             }
67         }
68     }
69     return 0;
70 }
```

Here is the call graph for this function:




Here is the caller graph for this function:



#### 5.11.3.3 getch()

```
int getch (
            void  )
```

Definition at line 14 of file getch.c.

```
15 {
16              char   ch;
17              int   error;
18              static struct termios Otty, Ntty;
19
20              fflush(stdout);
21              tcgetattr(0, &Otty);
22              Ntty = Otty;
23              Ntty.c_iflag  =  0;
24              Ntty.c_oflag  =  0;
25              Ntty.c_lflag &= ~ICANON;
26 #if 1
27              Ntty.c_lflag &= ~ECHO;
28 #else
29              Ntty.c_lflag |=  ECHO;
30 #endif
31              Ntty.c_cc[VMIN]  = CCHAR;
32              Ntty.c_cc[VTIME] = CTIME;
33
34 #if 1
35 #define FLAG TCSAFLUSH
36 #else
37 #define FLAG TCSANOW
38 #endif
39
40              if (0 == (error = tcsetattr(0, FLAG, &Ntty)))
41              {
42                        error  = read(0, &ch, 1 );
43                        error += tcsetattr(0, FLAG, &Otty);
44              }
45
46              return (error == 1 ? (int) ch : -1 );
47 }
```

#### 5.11.3.4 init_tetris_table()

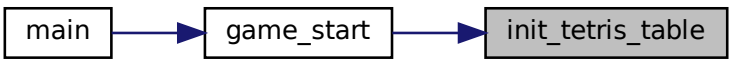
```
int init_tetris_table (
            void  )
```

.

Definition at line 75 of file tetrisV01.c.

```
76 {
77     int i = 0, j = 0;
78
79     for (i = 0; i < 20; i++)
80         for (j = 1; j < 9; j++)
81             tetris_table[i][j] = 0;
82
83     for (i = 0; i < 21; i++)
84     {
85         tetris_table[i][0] = 1;
86         tetris_table[i][9] = 1;
87     }
88
89     for (j = 1; j < 9; j++)
90         tetris_table[20][j] = 1;
91
92     return 0;
93 }
```

Here is the caller graph for this function:



#### 5.11.3.5 print_result()

```
int print_result (
            void  )
```

Definition at line 98 of file result.c.

```
99 {
100     //
101     char *home_dir = getenv("HOME");
102     char result_file[256];
103     sprintf(result_file, "%s/result", home_dir);
104 
105     FILE *fp = NULL;
106     char ch = 1;
107 
108     fp = fopen(result_file, "rb");
109 
110     if (fp == NULL)
111         return 0;
112 
113     system("clear");
114 
115     printf("\n\t\t\t\tText Tetris");
116     printf("\n\t\t\t\t Game Stats\n\n");
117     printf("\n\t\tName\t\tScore\t   Date\t\t Time");
118 
119     while (1)
120     {
121         fread(&temp_result, sizeof(struct result), 1, fp);
122         if (!feof(fp))
123         {
124             printf("\n\t=======================================================");
125             printf("\n\t\t%s\n\t\t\t\t%ld\t %d.%d.%d.  |  %d:%d\n", temp_result.name, temp_result.point,
      temp_result.year, temp_result.month, temp_result.day, temp_result.hour, temp_result.min);
126         }
127         else
128         {
129             break;
130         }
131     }
132 
133     fclose(fp);
134 
135     printf("\n\n\tBack to the game menu : M");
136     while (1)
137     {
138         ch = getch();
139         if (ch == 77 || ch == 109)
140             break;
141     }
142     return 0;
143 }
```

#### 5.11.3.6 save_result()

```
int save_result (
            long point,
            int best_point )
```

Definition at line 4 of file result.c.

```
5 {
6     //
7     char *home_dir = getenv("HOME");
8     char result_file[256];
9     sprintf(result_file, "%s/result", home_dir);
10 
11     FILE *fp = NULL;
12     time_t ptime;
13     struct tm *t;
14     //
15     printf("\n\n Final score : %ld ", point);
16     printf("\n\n Please enter your name : ");
17     scanf("%s%*c", temp_result.name);
18     temp_result.point = point;
19 
20     if (temp_result.point >= best_point)
21         best_point = temp_result.point;
22 
23     ptime = time(NULL);    //
24     t = localtime(&ptime); //
```

```
25
26      temp_result.year = t->tm_year + 1900;
27      temp_result.month = t->tm_mon + 1;
28      temp_result.day = t->tm_mday;
29      temp_result.hour = t->tm_hour;
30      temp_result.min = t->tm_min;
31
32      fp = fopen(result_file, "ab");
33      fseek(fp, 1, SEEK_END);
34      fwrite(&temp_result, sizeof(struct result), 1, fp);
35      fclose(fp);
36      return 0;
37  }
```

#### 5.11.3.7 search_result()

```
int search_result (
            void  )
```

Definition at line 39 of file result.c.

```
40  {
41      //
42      char *home_dir = getenv("HOME");
43      char result_file[256];
44      sprintf(result_file, "%s/result", home_dir);
45
46      FILE *fp = NULL;
47      char name[30];
48      char ch;
49      int find = 0;
50
51      fp = fopen(result_file, "rb");
52
53      if (fp == NULL)
54          return 0;
55
56      system("clear");
57
58      printf("\n\n\t\tEnter the name your to search.  : ");
59      scanf("%s%*c", name);
60
61      printf("\n\t\t\t\tText Tetris");
62      printf("\n\t\t\t\t Game Stats\n\n");
63      printf("\n\t\tName\t\tScore\t   Date\t\t Time");
64
65      while (1)
66      {
67          fread(&temp_result, sizeof(struct result), 1, fp);
68          if (!feof(fp))
69          {
70              if (!strcmp(temp_result.name, name))
71              {
72                  find = 1;
73                  printf("\n\t=========================================================");
74                  printf("\n\t\t%s\n\t\t\t\t%ld\t%d.%d.%d.  |  %d:%d\n", temp_result.name,
    temp_result.point, temp_result.year, temp_result.month, temp_result.day, temp_result.hour,
    temp_result.min);
75              }
76          }
77          else
78          {
79              break;
80          }
81      }
82
83      if (find == 0)
84          printf("\n\n\n\t\tThis name is not found.");
85
86      printf("\n\n\n\t\tBack to the game menu : M");
87      while (1)
88      {
89          ch = getch();
90          if (ch == 77 || ch == 109)
91              break;
92      }
93
94      return 0;
95  }
```

#### 5.11.3.8 update()

```
int update (
            int signum )
```

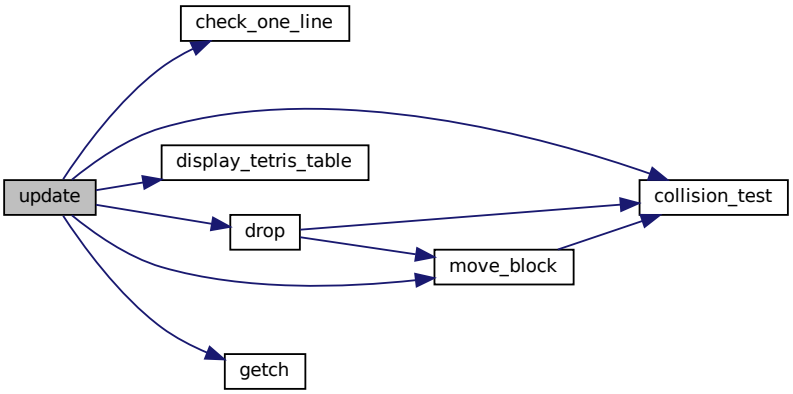
. ,,

Definition at line 6 of file update.c.

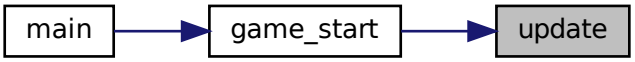
```
7 {
8     static int downcount = 0;
9     static int setcount = 0;
10     static long speedcount = 0;
11     static int countrange = 5;
12     static int firststart = 0;
13
14     char ch;
15
16     srand((unsigned)time(NULL));
17
18     if (firststart == 0)
19     {
20         block_number = rand() % 7;
21         if (firststart == 0)
22             firststart++;
23     }
24
25     display_tetris_table(&countrange);
26     check_one_line();
27
28     if (downcount == countrange - 1)
29     {
30         point += 1;
31         move_block(DOWN);
32     }
33
34     if (speedcount == 499)
35     {
36         if (countrange != 1)
37             countrange--;
38     }
39
40     downcount++;
41     downcount %= countrange;
42     speedcount++;
43     speedcount %= 500;
44
45     if (x == 3 && y == 0)
46     {
47         if (collision_test(LEFT) || collision_test(RIGHT) || collision_test(DOWN) ||
       collision_test(ROTATE))
48         {
49             printf("\n Game End! \n");
50             downcount = 0;
51             setcount = 0;
52             speedcount = 0;
53             countrange = 5;
54             firststart = 0;
55             game = GAME_END;
56         }
57     }
58
59     if (collision_test(DOWN))
60     {
61         if (setcount == 9)
62         {
63             block_number = next_block_number;
64             next_block_number = rand() % 7;
65             block_state = 0;
66             x = 3;
67             y = 0;
68         }
69         setcount++;
70         setcount %= 10;
71     }
72
73     ch = getch();
74
75     switch (ch)
76     {
77     case 74:
78     case 106:
79         move_block(LEFT);
80         break;
81     case 76:
82     case 108:
83         move_block(RIGHT);
84         break;
```

```
85      case 75:
86      case 107:
87          move_block(DOWN);
88          break;
89      case 73:
90      case 105:
91          move_block(ROTATE);
92          break;
93      case 65:
94      case 97:
95          drop();
96          break;
97      case 80:
98      case 112:
99          downcount = 0;
100          setcount = 0;
101          speedcount = 0;
102          countrange = 5;
103          firststart = 0;
104          game = GAME_END;
105          break;
106      default:
107          break;
108      }
109      return 0;
110  }
```

Here is the call graph for this function:




Here is the caller graph for this function:




## 5.11.4 Variable Documentation

### 5.11.4.1 best_point

```
int best_point = 0
```

Definition at line 47 of file tetrisV01.h.

#### 5.11.4.2 game

```
int game = GAME_END
```
Definition at line 46 of file tetrisV01.h.

#### 5.11.4.3 point

```
long point = 0
```
Definition at line 48 of file tetrisV01.h.

#### 5.11.4.4 tetris_table

```
char tetris_table[21][10]
```
Definition at line 43 of file tetrisV01.h.

#### 5.11.4.5 x

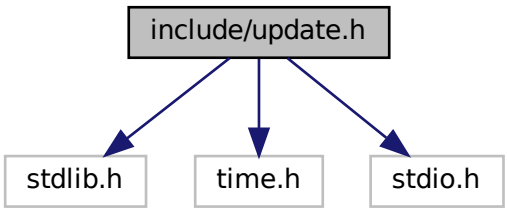
```
int x = 3
```
Definition at line 45 of file tetrisV01.h.

#### 5.11.4.6 y

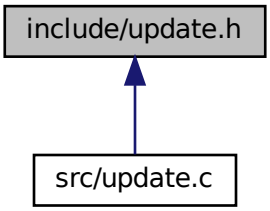
```
int y = 0
```
Definition at line 45 of file tetrisV01.h.

## 5.12 include/update.h File Reference

```
#include <stdlib.h>
#include <time.h>
#include <stdio.h>
```
Include dependency graph for update.h:



This graph shows which files directly or indirectly include this file:




## Macros

- #define GAME_START 0
- #define GAME_END 1

## Enumerations

- enum Direction {
  LEFT , RIGHT , DOWN , ROTATE ,
  LEFT , RIGHT , DOWN , ROTATE }
- enum BlockType {
  I_BLOCK , T_BLOCK , S_BLOCK , Z_BLOCK ,
  L_BLOCK , J_BLOCK , O_BLOCK , I_BLOCK ,
  T_BLOCK , S_BLOCK , Z_BLOCK , L_BLOCK ,
  J_BLOCK , O_BLOCK , I_BLOCK , T_BLOCK ,
  S_BLOCK , Z_BLOCK , L_BLOCK , J_BLOCK ,
  O_BLOCK }

## Functions

- int display_tetris_table (int *)
- int move_block (int)
- int drop (void)
- int collision_test (int)
- int check_one_line (void)
- int getch (void)

## Variables

- char i_block [4][4][4]
- char t_block [4][4][4]
- char s_block [4][4][4]
- char z_block [4][4][4]
- char l_block [4][4][4]
- char j_block [4][4][4]
- char o_block [4][4][4]
- int block_number = 0
- int next_block_number = 0
- int block_state = 0
- char tetris_table [21][10]

- int x
- int y
- int game
- int best_point
- long point

### 5.12.1 Macro Definition Documentation

#### 5.12.1.1 GAME_END

```
#define GAME_END 1
```

Definition at line 9 of file update.h.

#### 5.12.1.2 GAME_START

```
#define GAME_START 0
```

Definition at line 8 of file update.h.

### 5.12.2 Enumeration Type Documentation

#### 5.12.2.1 BlockType

```
enum BlockType
```

**Enumerator**

| | |
|---|---|
| I_BLOCK | |
| T_BLOCK | |
| S_BLOCK | |
| Z_BLOCK | |
| L_BLOCK | |
| J_BLOCK | |
| O_BLOCK | |
| I_BLOCK | |
| T_BLOCK | |
| S_BLOCK | |
| Z_BLOCK | |
| L_BLOCK | |
| J_BLOCK | |
| O_BLOCK | |
| I_BLOCK | |
| T_BLOCK | |
| S_BLOCK | |
| Z_BLOCK | |
| L_BLOCK | |
| J_BLOCK | |
| O_BLOCK | |

Definition at line 20 of file update.h.

```
21 {
22     I_BLOCK,
```

```
23     T_BLOCK,
24     S_BLOCK,
25     Z_BLOCK,
26     L_BLOCK,
27     J_BLOCK,
28     O_BLOCK
29 };
```

#### 5.12.2.2 Direction

```
enum Direction
```

**Enumerator**

| | |
|---|---|
| LEFT | |
| RIGHT | |
| DOWN | |
| ROTATE | |
| LEFT | |
| RIGHT | |
| DOWN | |
| ROTATE | |

Definition at line 12 of file update.h.

```
13 {
14     LEFT,
15     RIGHT,
16     DOWN,
17     ROTATE
18 };
```

### 5.12.3 Function Documentation

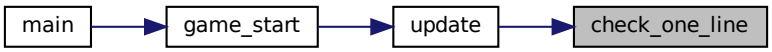
#### 5.12.3.1 check_one_line()

```
int check_one_line (
            void  )
```

Definition at line 300 of file update.c.

```
301 {
302     int i, j;
303     int ti, tj;
304     int line_hole;
305 
306     for (i = 19; i > 0; i--)
307     {
308         line_hole = 0;
309         for (j = 1; j < 9; j++)
310         {
311             if (tetris_table[i][j] == 0)
312             {
313                 line_hole = 1;
314             }
315         }
316 
317         if (line_hole == 0)
318         {
319             point += 1000;
320             for (ti = i; ti > 0; ti--)
321             {
322                 for (tj = 0; tj < 9; tj++)
323                 {
324                     tetris_table[ti][tj] = tetris_table[ti - 1][tj];
325                 }
326             }
327         }
328     }
329 
330     return 0;
331 }
```

Here is the caller graph for this function:




#### 5.12.3.2 collision_test()

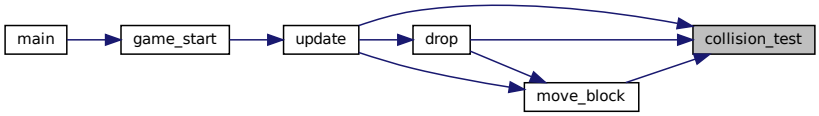
```
int collision_test (
            int command )
```

Definition at line 202 of file update.c.

```
203 {
204     int i, j;
205     int tempx, tempy;
206     int oldx, oldy;
207     int temp_block_state;
208     char(*block_pointer)[4][4][4];
209     char temp_tetris_table[21][10];
210 
211     oldx = tempx = x;
212     oldy = tempy = y;
213     temp_block_state = block_state;
214 
215     switch (command)
216     {
217     case LEFT:
218         tempx--;
219         break;
220     case RIGHT:
221         tempx++;
222         break;
223     case DOWN:
224         tempy++;
225         break;
226     case ROTATE:
227         temp_block_state++;
228         temp_block_state %= 4;
229         break;
230     }
231 
232     switch (block_number)
233     {
234     case I_BLOCK:
235         block_pointer = &i_block;
236         break;
237     case T_BLOCK:
238         block_pointer = &t_block;
239         break;
240     case S_BLOCK:
241         block_pointer = &s_block;
242         break;
243     case Z_BLOCK:
244         block_pointer = &z_block;
245         break;
246     case L_BLOCK:
247         block_pointer = &l_block;
248         break;
249     case J_BLOCK:
250         block_pointer = &j_block;
251         break;
252     case O_BLOCK:
253         block_pointer = &o_block;
254         break;
255     }
256 
257     for (i = 0; i < 21; i++)
258     {
259         for (j = 0; j < 10; j++)
260         {
261             temp_tetris_table[i][j] = tetris_table[i][j];
262         }
263     }
264 
```

```
265     for (i = 0, oldy = y; i < 4; i++, oldy++)
266     {
267         for (j = 0, oldx = x; j < 4; j++, oldx++)
268         {
269             if (oldx > 0 && oldx < 9 && oldy < 20 && oldy > 0)
270             {
271                 if ((*block_pointer)[block_state][i][j] == 1)
272                     temp_tetris_table[oldy][oldx] = 0;
273             }
274         }
275     }
276 
277     for (i = 0; i < 4; i++)
278     {
279         for (j = 0; j < 4; j++)
280         {
281 
282             if (temp_tetris_table[tempy + i][tempx + j] == 1 && (*block_pointer)[temp_block_state][i][j]
      == 1)
283                 return 1;
284         }
285     }
286 
287     return 0;
288 }
```

Here is the caller graph for this function:




#### 5.12.3.3 display_tetris_table()

```
int display_tetris_table (
            int * countrange )
```

Definition at line 4 of file display_tetris.c.

```
5 {
6     int i, j;
7     char(*block_pointer)[4][4][4] = NULL;
8 
9     switch (next_block_number)
10     {
11     case I_BLOCK:
12         block_pointer = &i_block;
13         break;
14     case T_BLOCK:
15         block_pointer = &t_block;
16         break;
17     case S_BLOCK:
18         block_pointer = &s_block;
19         break;
20     case Z_BLOCK:
21         block_pointer = &z_block;
22         break;
23     case L_BLOCK:
24         block_pointer = &l_block;
25         break;
26     case J_BLOCK:
27         block_pointer = &j_block;
28         break;
29     case O_BLOCK:
30         block_pointer = &o_block;
31         break;
32     }
33 
34     system("clear");
35     printf("\n Score: %ld | Speed: %d | hihgest score: %d", point, *countrange, best_point);
36 
37     printf("\n\n Next Block\n");
38 
39     for (i = 0; i < 4; i++)
```

```
40     {
41         printf("\n ");
42         for (j = 0; j < 4; j++)
43         {
44             if ((*block_pointer)[0][i][j] == 1)
45                 printf("");
46             else if ((*block_pointer)[0][i][j] == 0)
47                 printf(" ");
48         }
49     }
50
51     for (i = 2; i < 21; i++)
52     {
53         printf("\t");
54         for (j = 0; j < 10; j++)
55         {
56             if (j == 0 || j == 9 || (i == 20 && (j > 1 || j < 9)))
57             {
58                 printf("");
59             }
60             else if (tetris_table[i][j] == 1)
61                 printf("");
62             else if (tetris_table[i][j] == 0)
63                 printf(" ");
64         }
65         printf("\n");
66     }
67     printf("\n GAME STOP : P");
68     return 0;
69 }
```

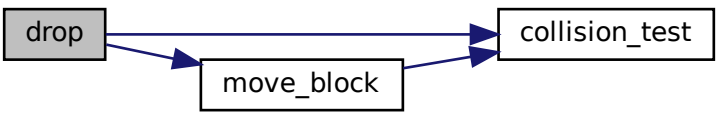
#### 5.12.3.4 drop()

```
int drop (
              void  )
```

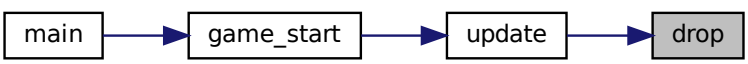
Definition at line 291 of file update.c.

```
292 {
293     while (!collision_test(DOWN))
294         move_block(DOWN);
295
296     return 0;
297 }
```

Here is the call graph for this function:




Here is the caller graph for this function:



#### 5.12.3.5 getch()

```
int getch (
            void  )
```

Definition at line 14 of file getch.c.

```
15 {
16              char   ch;
17              int   error;
18              static struct termios Otty, Ntty;
19
20              fflush(stdout);
21              tcgetattr(0, &Otty);
22              Ntty = Otty;
23              Ntty.c_iflag  =  0;
24              Ntty.c_oflag  =  0;
25              Ntty.c_lflag &= ~ICANON;
26 #if 1
27              Ntty.c_lflag &= ~ECHO;
28 #else
29              Ntty.c_lflag |=  ECHO;
30 #endif
31              Ntty.c_cc[VMIN]  = CCHAR;
32              Ntty.c_cc[VTIME] = CTIME;
33
34 #if 1
35 #define FLAG TCSAFLUSH
36 #else
37 #define FLAG TCSANOW
38 #endif
39
40              if (0 == (error = tcsetattr(0, FLAG, &Ntty)))
41              {
42                        error  = read(0, &ch, 1 );
43                        error += tcsetattr(0, FLAG, &Otty);
44              }
45
46              return (error == 1 ? (int) ch : -1 );
47 }
```

#### 5.12.3.6 move_block()

```
int move_block (
            int command )
```

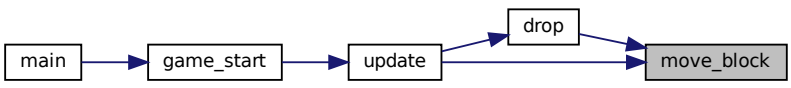
Definition at line 113 of file update.c.

```
114 {
115     int i, j;
116     int newx, newy;
117     int oldx, oldy;
118     int old_block_state;
119     char(*block_pointer)[4][4][4] = NULL;
120
121     newx = x;
122     newy = y;
123
124     old_block_state = block_state;
125
126     if (collision_test(command) == 0)
127     {
128         switch (command)
129         {
130         case LEFT:
131             newx--;
132             break;
133         case RIGHT:
134             newx++;
135             break;
136         case DOWN:
137             newy++;
138             break;
139         case ROTATE:
140             block_state++;
141             block_state %= 4;
142             break;
143         }
144     }
145     else
146     {
147         return 1;
148     }
149
150     switch (block_number)
151     {
```

```
152     case I_BLOCK:
153         block_pointer = &i_block;
154         break;
155     case T_BLOCK:
156         block_pointer = &t_block;
157         break;
158     case S_BLOCK:
159         block_pointer = &s_block;
160         break;
161     case Z_BLOCK:
162         block_pointer = &z_block;
163         break;
164     case L_BLOCK:
165         block_pointer = &l_block;
166         break;
167     case J_BLOCK:
168         block_pointer = &j_block;
169         break;
170     case O_BLOCK:
171         block_pointer = &o_block;
172         break;
173     }
174 
175     for (i = 0, oldy = y; i < 4; i++, oldy++)
176     {
177         for (j = 0, oldx = x; j < 4; j++, oldx++)
178         {
179             if (oldx > 0 && oldx < 9 && oldy < 20 && oldy > 0)
180                 if ((*block_pointer)[old_block_state][i][j] == 1)
181                     tetris_table[oldy][oldx] = 0;
182         }
183     }
184 
185     x = newx;
186     y = newy;
187 
188     for (i = 0, newy = y; i < 4; i++, newy++)
189     {
190         for (j = 0, newx = x; j < 4; j++, newx++)
191         {
192             if (newx > 0 && newx < 9 && newy < 20 && newy > 0)
193                 if ((*block_pointer)[block_state][i][j] == 1)
194                     tetris_table[newy][newx] = (*block_pointer)[block_state][i][j];
195         }
196     }
197 
198     return 0;
199 }
```

Here is the call graph for this function:


Here is the caller graph for this function:




### 5.12.4 Variable Documentation

#### 5.12.4.1 best_point

```
int best_point  [extern]
```

Definition at line 47 of file tetrisV01.h.

#### 5.12.4.2 block_number

```
int block_number = 0
```

Definition at line 174 of file update.h.

#### 5.12.4.3 block_state

```
int block_state = 0
```

Definition at line 176 of file update.h.

#### 5.12.4.4 game

```
int game  [extern]
```

Definition at line 46 of file tetrisV01.h.

#### 5.12.4.5 i_block

```
char i_block[4][4][4]
```

**Initial value:**

```
=
    {
        {{1, 1, 1, 1},
         {0, 0, 0, 0},
         {0, 0, 0, 0},
         {0, 0, 0, 0}},
        {{0, 0, 0, 1},
         {0, 0, 0, 1},
         {0, 0, 0, 1},
         {0, 0, 0, 1}},
        {{0, 0, 0, 0},
         {0, 0, 0, 0},
         {1, 1, 1, 1},
         {0, 0, 0, 0}},
        {{1, 0, 0, 0},
         {1, 0, 0, 0},
         {1, 0, 0, 0},
         {1, 0, 0, 0}}}
```

Definition at line 40 of file update.h.

#### 5.12.4.6 j_block

```
char j_block[4][4][4]
```

**Initial value:**

```
=
    {
        {{0, 1, 0, 0},
         {0, 1, 0, 0},
         {1, 1, 0, 0},
         {0, 0, 0, 0}},
        {{1, 0, 0, 0},
         {1, 1, 1, 0},
         {0, 0, 0, 0},
         {0, 0, 0, 0}},
        {{1, 1, 0, 0},
         {1, 0, 0, 0},
         {1, 0, 0, 0},
         {0, 0, 0, 0}},
        {{1, 1, 1, 0},
         {0, 0, 1, 0},
         {0, 0, 0, 0},
         {0, 0, 0, 0}}}
```

Definition at line 135 of file update.h.

#### 5.12.4.7 l_block

```
char l_block[4][4][4]
```

**Initial value:**

```
=
    {
        {{1, 0, 0, 0},
         {1, 0, 0, 0},
         {1, 1, 0, 0},
         {0, 0, 0, 0}},
        {{1, 1, 1, 0},
         {1, 0, 0, 0},
         {0, 0, 0, 0},
         {0, 0, 0, 0}},
        {{1, 1, 0, 0},
         {0, 1, 0, 0},
         {0, 1, 0, 0},
         {0, 0, 0, 0}},
        {{0, 0, 1, 0},
         {1, 1, 1, 0},
         {0, 0, 0, 0},
         {0, 0, 0, 0}}}
```

Definition at line 116 of file update.h.

#### 5.12.4.8 next_block_number

```
int next_block_number = 0
```

Definition at line 175 of file update.h.

#### 5.12.4.9 o_block

```
char o_block[4][4][4]
```

**Initial value:**

```
=
    {
        {{1, 1, 0, 0},
         {1, 1, 0, 0},
         {0, 0, 0, 0},
         {0, 0, 0, 0}},
        {{1, 1, 0, 0},
         {1, 1, 0, 0},
         {0, 0, 0, 0},
         {0, 0, 0, 0}},
        {{1, 1, 0, 0},
         {1, 1, 0, 0},
         {0, 0, 0, 0},
         {0, 0, 0, 0}},
        {{1, 1, 0, 0},
         {1, 1, 0, 0},
         {0, 0, 0, 0},
         {0, 0, 0, 0}}}
```

Definition at line 154 of file update.h.

#### 5.12.4.10 point

```
long point  [extern]
```

Definition at line 48 of file tetrisV01.h.

#### 5.12.4.11 s_block

```
char s_block[4][4][4]
```

**Initial value:**

```
=
    {
        {{1, 0, 0, 0},
```

```
         {1, 1, 0, 0},
         {0, 1, 0, 0},
         {0, 0, 0, 0}},
        {{0, 1, 1, 0},
         {1, 1, 0, 0},
         {0, 0, 0, 0},
         {0, 0, 0, 0}},
        {{0, 1, 0, 0},
         {0, 1, 1, 0},
         {0, 0, 1, 0},
         {0, 0, 0, 0}},
        {{1, 1, 0, 0},
         {0, 1, 1, 0},
         {0, 0, 0, 0},
         {0, 0, 0, 0}}}
```

Definition at line 78 of file update.h.

#### 5.12.4.12 t_block

```
char t_block[4][4][4]
```

**Initial value:**

```
=
    {
        {{1, 0, 0, 0},
         {1, 1, 0, 0},
         {1, 0, 0, 0},
         {0, 0, 0, 0}},
        {{0, 1, 0, 0},
         {1, 1, 0, 0},
         {0, 1, 0, 0},
         {0, 0, 0, 0}},
        {{0, 0, 0, 0},
         {1, 1, 1, 0},
         {0, 1, 0, 0},
         {0, 0, 0, 0}},
        {{0, 1, 0, 0},
         {1, 1, 0, 0},
         {0, 1, 0, 0},
         {0, 0, 0, 0}}}
```

Definition at line 59 of file update.h.

#### 5.12.4.13 tetris_table

```
char tetris_table[21][10]  [extern]
```

Definition at line 43 of file tetrisV01.h.

#### 5.12.4.14 x

```
int x  [extern]
```

Definition at line 45 of file tetrisV01.h.

#### 5.12.4.15 y

```
int y
```

Definition at line 181 of file update.h.

#### 5.12.4.16 z_block

```
char z_block[4][4][4]
```

**Initial value:**

```
=
    {
        {{0, 1, 0, 0},
         {1, 1, 0, 0},
         {1, 0, 0, 0},
         {0, 0, 0, 0}},
        {{1, 1, 0, 0},
```

```
        {0, 1, 1, 0},
        {0, 0, 0, 0},
        {0, 0, 0, 0}},
       {{0, 0, 1, 0},
        {0, 1, 1, 0},
        {0, 1, 0, 0},
        {0, 0, 0, 0}},
       {{1, 1, 0, 0},
        {1, 0, 1, 0},
        {0, 0, 0, 0},
        {0, 0, 0, 0}}}
```

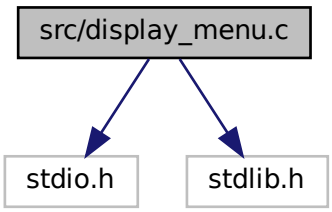
Definition at line 97 of file update.h.

## 5.13 README.md File Reference

## 5.14 src/display_menu.c File Reference

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

Include dependency graph for display_menu.c:




### Functions

- int display_menu (void)

### 5.14.1 Function Documentation

#### 5.14.1.1 display_menu()

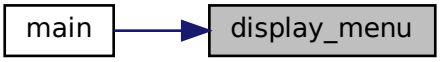
```
int display_menu (
            void  )
```

Definition at line 5 of file display_menu.c.

```
6 {
7     int menu = 0;
8
9     while(1)
10     {
11         system("clear");
12         printf("\n\n\t\t\t\tText Tetris");
13         printf("\n\t\t\t============================");
14         printf("\n\t\t\t\tGAME MENU\t\n");
15         printf("\n\t\t\t============================");
16         printf("\n\t\t\t   1) Game Start");
17         printf("\n\t\t\t   2) Search history");
18         printf("\n\t\t\t   3) Record Output");
19         printf("\n\t\t\t   4) QUIT");
20         printf("\n\t\t\t============================");
21         printf("\n\t\t\t\t\t SELECT : ");
22         scanf("%d",&menu);
23         if(menu < 1 || menu > 4)
```

```
24         {
25             continue;
26         }
27         else
28         {
29             return menu;
30         }
31     }
32     return 0;
33 }
```

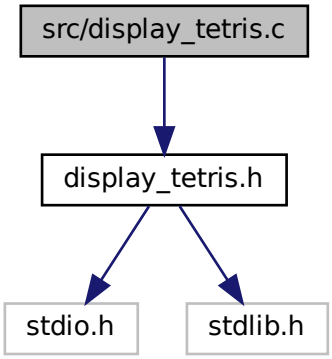
Here is the caller graph for this function:




## 5.15 src/display_tetris.c File Reference

```
#include "display_tetris.h"
```

Include dependency graph for display_tetris.c:




### Functions

- int display_tetris_table (int *countrange)

### 5.15.1 Function Documentation

#### 5.15.1.1 display_tetris_table()

```
int display_tetris_table (
            int * countrange )
```

Definition at line 4 of file display_tetris.c.

```
5 {
```

```
6       int i, j;
7       char(*block_pointer)[4][4][4] = NULL;
8
9       switch (next_block_number)
10       {
11      case I_BLOCK:
12          block_pointer = &i_block;
13          break;
14      case T_BLOCK:
15          block_pointer = &t_block;
16          break;
17      case S_BLOCK:
18          block_pointer = &s_block;
19          break;
20      case Z_BLOCK:
21          block_pointer = &z_block;
22          break;
23      case L_BLOCK:
24          block_pointer = &l_block;
25          break;
26      case J_BLOCK:
27          block_pointer = &j_block;
28          break;
29      case O_BLOCK:
30          block_pointer = &o_block;
31          break;
32      }
33
34      system("clear");
35      printf("\n Score: %ld | Speed: %d | hihgest score: %d", point, *countrange, best_point);
36
37      printf("\n\n Next Block\n");
38
39      for (i = 0; i < 4; i++)
40      {
41          printf("\n ");
42          for (j = 0; j < 4; j++)
43          {
44              if ((*block_pointer)[0][i][j] == 1)
45                  printf("");
46              else if ((*block_pointer)[0][i][j] == 0)
47                  printf(" ");
48          }
49      }
50
51      for (i = 2; i < 21; i++)
52      {
53          printf("\t");
54          for (j = 0; j < 10; j++)
55          {
56              if (j == 0 || j == 9 || (i == 20 && (j > 1 || j < 9)))
57              {
58                  printf("");
59              }
60              else if (tetris_table[i][j] == 1)
61                  printf("");
62              else if (tetris_table[i][j] == 0)
63                  printf(" ");
64          }
65          printf("\n");
66      }
67      printf("\n GAME STOP : P");
68      return 0;
69  }
```

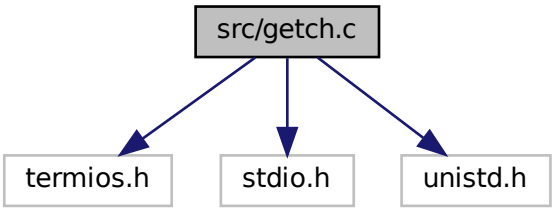
Here is the caller graph for this function:


## 5.16 src/getch.c File Reference

```
#include <termios.h>
#include <stdio.h>
```

```
#include <unistd.h>
```

Include dependency graph for getch.c:




## Macros

- #define CCHAR 0
- #define CTIME 1
- #define FLAG TCSAFLUSH

## Functions

- int getch (void)

### 5.16.1 Macro Definition Documentation

#### 5.16.1.1 CCHAR

```
#define CCHAR 0
```

Definition at line 7 of file getch.c.

#### 5.16.1.2 CTIME

```
#define CTIME 1
```

Definition at line 11 of file getch.c.

#### 5.16.1.3 FLAG

```
#define FLAG TCSAFLUSH
```

### 5.16.2 Function Documentation

#### 5.16.2.1 getch()

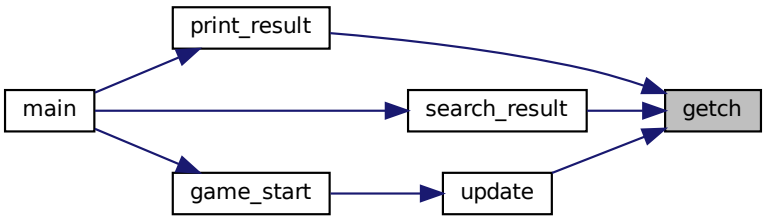
```
int getch (
            void  )
```

Definition at line 14 of file getch.c.

```
15 {
16             char   ch;
17             int   error;
```

```
18              static struct termios Otty, Ntty;
19
20              fflush(stdout);
21              tcgetattr(0, &Otty);
22              Ntty = Otty;
23              Ntty.c_iflag  =  0;
24              Ntty.c_oflag  =  0;
25              Ntty.c_lflag &= ~ICANON;
26 #if 1
27             Ntty.c_lflag &= ~ECHO;
28 #else
29             Ntty.c_lflag |=  ECHO;
30 #endif
31             Ntty.c_cc[VMIN]  = CCHAR;
32             Ntty.c_cc[VTIME] = CTIME;
33
34 #if 1
35 #define FLAG TCSAFLUSH
36 #else
37 #define FLAG TCSANOW
38 #endif
39
40             if (0 == (error = tcsetattr(0, FLAG, &Ntty)))
41             {
42                       error  = read(0, &ch, 1 );
43                       error += tcsetattr(0, FLAG, &Otty);
44             }
45
46             return (error == 1 ? (int) ch : -1 );
47 }
```

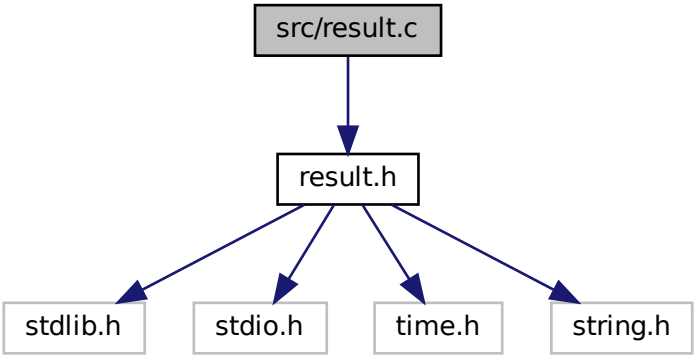
Here is the caller graph for this function:




## 5.17 src/result.c File Reference

```
#include "result.h"
```

Include dependency graph for result.c:




## Functions

- int save_result (long point, int best_point)
- int search_result (void)
- int print_result (void)

### 5.17.1 Function Documentation

#### 5.17.1.1 print_result()

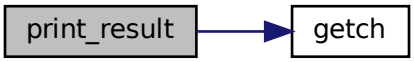
```
int print_result (
            void  )
```

Definition at line 98 of file result.c.

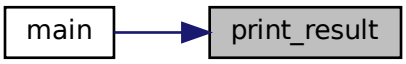
```
99 {
100     //
101     char *home_dir = getenv("HOME");
102     char result_file[256];
103     sprintf(result_file, "%s/result", home_dir);
104 
105     FILE *fp = NULL;
106     char ch = 1;
107 
108     fp = fopen(result_file, "rb");
109 
110     if (fp == NULL)
111         return 0;
112 
113     system("clear");
114 
115     printf("\n\t\t\t\tText Tetris");
116     printf("\n\t\t\t\t Game Stats\n\n");
117     printf("\n\t\tName\t\tScore\t   Date\t\t Time");
118 
119     while (1)
120     {
121         fread(&temp_result, sizeof(struct result), 1, fp);
122         if (!feof(fp))
123         {
124             printf("\n\t=======================================================");
125             printf("\n\t\t%s\n\t\t\t\t%ld\t %d.%d.%d.  |  %d:%d\n", temp_result.name, temp_result.point,
      temp_result.year, temp_result.month, temp_result.day, temp_result.hour, temp_result.min);
126         }
127         else
128         {
129             break;
130         }
```

```
131     }
132
133     fclose(fp);
134
135     printf("\n\n\tBack to the game menu : M");
136     while (1)
137     {
138         ch = getch();
139         if (ch == 77 || ch == 109)
140             break;
141     }
142     return 0;
143 }
```

Here is the call graph for this function:




Here is the caller graph for this function:




#### 5.17.1.2 save_result()

```
int save_result (
              long point,
              int best_point )
```

Definition at line 4 of file result.c.

```
5 {
6     //
7     char *home_dir = getenv("HOME");
8     char result_file[256];
9     sprintf(result_file, "%s/result", home_dir);
10
11     FILE *fp = NULL;
12     time_t ptime;
13     struct tm *t;
14     //
15     printf("\n\n Final score : %ld ", point);
16     printf("\n\n Please enter your name : ");
17     scanf("%s%*c", temp_result.name);
18     temp_result.point = point;
19
20     if (temp_result.point >= best_point)
21         best_point = temp_result.point;
22
23     ptime = time(NULL);    //
24     t = localtime(&ptime); //
25
26     temp_result.year = t->tm_year + 1900;
27     temp_result.month = t->tm_mon + 1;
28     temp_result.day = t->tm_mday;
29     temp_result.hour = t->tm_hour;
```

```
30     temp_result.min = t->tm_min;
31
32     fp = fopen(result_file, "ab");
33     fseek(fp, 1, SEEK_END);
34     fwrite(&temp_result, sizeof(struct result), 1, fp);
35     fclose(fp);
36     return 0;
37 }
```

Here is the caller graph for this function:


#### 5.17.1.3 search_result()

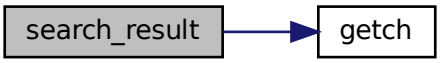
```
int search_result (
            void  )
```

Definition at line 39 of file result.c.

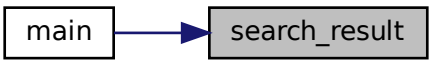
```
40 {
41     //
42     char *home_dir = getenv("HOME");
43     char result_file[256];
44     sprintf(result_file, "%s/result", home_dir);
45
46     FILE *fp = NULL;
47     char name[30];
48     char ch;
49     int find = 0;
50
51     fp = fopen(result_file, "rb");
52
53     if (fp == NULL)
54         return 0;
55
56     system("clear");
57
58     printf("\n\n\t\tEnter the name your to search.  : ");
59     scanf("%s%*c", name);
60
61     printf("\n\t\t\t\tText Tetris");
62     printf("\n\t\t\t\t Game Stats\n\n");
63     printf("\n\t\tName\t\tScore\t   Date\t\t Time");
64
65     while (1)
66     {
67         fread(&temp_result, sizeof(struct result), 1, fp);
68         if (!feof(fp))
69         {
70             if (!strcmp(temp_result.name, name))
71             {
72                 find = 1;
73                 printf("\n\t==========================================================");
74                 printf("\n\t\t%s\n\t\t\t\t%ld\t%d.%d.%d.  |  %d:%d\n", temp_result.name,
    temp_result.point, temp_result.year, temp_result.month, temp_result.day, temp_result.hour,
    temp_result.min);
75             }
76         }
77         else
78         {
79             break;
80         }
81     }
82
83     if (find == 0)
84         printf("\n\n\n\t\tThis name is not found.");
85
86     printf("\n\n\n\t\tBack to the game menu : M");
87     while (1)
88     {
```

```
89          ch = getch();
90          if (ch == 77 || ch == 109)
91              break;
92      }
93
94      return 0;
95  }
```

Here is the call graph for this function:



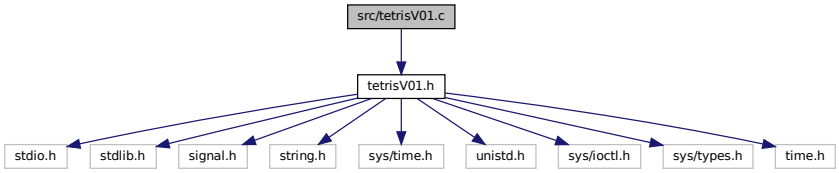

Here is the caller graph for this function:




## 5.18 src/tetrisV01.c File Reference

`#include "tetrisV01.h"`

Include dependency graph for tetrisV01.c:




### Functions

- int main (void)
- int game_start (void)
- int init_tetris_table (void)

### 5.18.1 Function Documentation

#### 5.18.1.1 game_start()

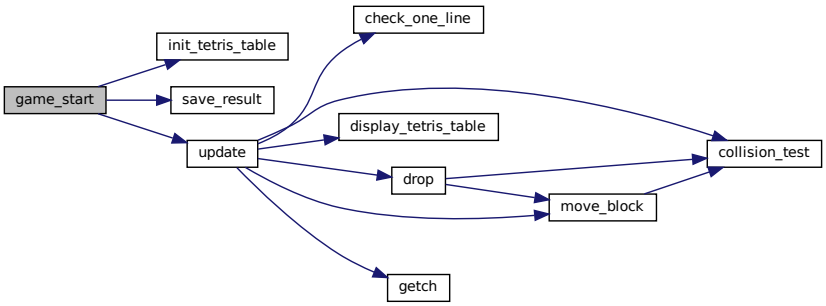
```
int game_start (
            void  )
```

. game . update() signal .

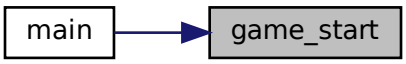
Definition at line 38 of file tetrisV01.c.

```
39 {
40
41     if (game == GAME_START)
42     {
43         init_tetris_table();
44
45         //
46         static struct itimerval timer;
47         //  type    . signal(SIGALRM, (void (*)(int))update);
48         signal(SIGVTALRM, (void (*)(int))update);
49         timer.it_value.tv_sec = 0;
50         timer.it_value.tv_usec = 1;
51         timer.it_interval.tv_sec = 0;
52         timer.it_interval.tv_usec = 1;
53         setitimer(ITIMER_VIRTUAL, &timer, NULL);
54
55         while (1)
56         {
57             if (game == GAME_END)
58             {
59                 signal(SIGALRM, SIG_IGN);
60                 save_result(point, best_point);
61
62                 x = 3, y = 0;
63                 point = 0;
64
65                 return 1;
66             }
67         }
68     }
69     return 0;
70 }
```

Here is the call graph for this function:




Here is the caller graph for this function:



#### 5.18.1.2 init_tetris_table()

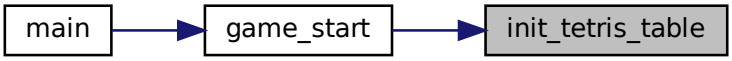
```
int init_tetris_table (
              void  )
```

.

Definition at line 75 of file tetrisV01.c.

```
76 {
77     int i = 0, j = 0;
78 
79     for (i = 0; i < 20; i++)
80         for (j = 1; j < 9; j++)
81             tetris_table[i][j] = 0;
82 
83     for (i = 0; i < 21; i++)
84     {
85         tetris_table[i][0] = 1;
86         tetris_table[i][9] = 1;
87     }
88 
89     for (j = 1; j < 9; j++)
90         tetris_table[20][j] = 1;
91 
92     return 0;
93 }
```

Here is the caller graph for this function:



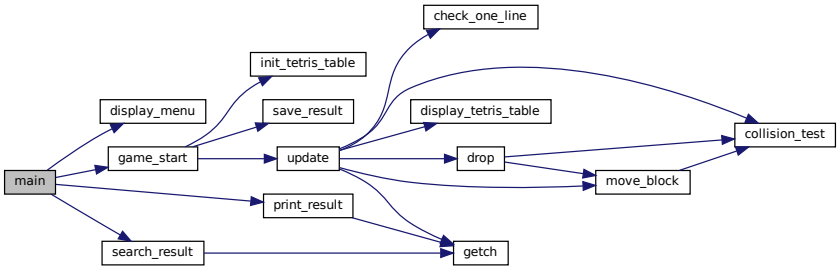

#### 5.18.1.3 main()

```
int main (
              void  )
```

Definition at line 3 of file tetrisV01.c.

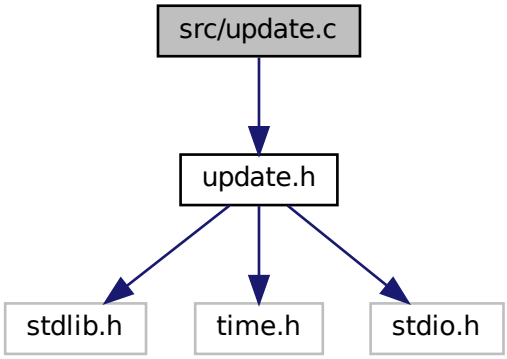
```
4 {
5     int menu = 1;
6 
7     while (menu)
8     {
9         menu = display_menu();
10 
11         switch (menu)
12         {
13         case 1:
14             game = GAME_START;
15             menu = game_start();
16             break;
17         case 2:
18             search_result();
19             break;
20         case 3:
21             print_result();
22             break;
23         case 4:
24             exit(0);
25             break;
26         default:
27             break;
28         }
29     }
30 
31     return 0;
32 }
```

Here is the call graph for this function:




## 5.19 src/update.c File Reference

`#include "update.h"`
Include dependency graph for update.c:




### Functions

- int update (int signum)
- int move_block (int command)
- int collision_test (int command)
- int drop (void)
- int check_one_line (void)

### 5.19.1 Function Documentation

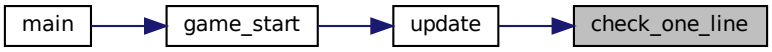
#### 5.19.1.1 check_one_line()

```
int check_one_line (
            void  )
```

Definition at line 300 of file update.c.
`301 {`

```
302     int i, j;
303     int ti, tj;
304     int line_hole;
305
306     for (i = 19; i > 0; i--)
307     {
308         line_hole = 0;
309         for (j = 1; j < 9; j++)
310         {
311             if (tetris_table[i][j] == 0)
312             {
313                 line_hole = 1;
314             }
315         }
316
317         if (line_hole == 0)
318         {
319             point += 1000;
320             for (ti = i; ti > 0; ti--)
321             {
322                 for (tj = 0; tj < 9; tj++)
323                 {
324                     tetris_table[ti][tj] = tetris_table[ti - 1][tj];
325                 }
326             }
327         }
328     }
329
330     return 0;
331 }
```

Here is the caller graph for this function:




#### 5.19.1.2 collision_test()

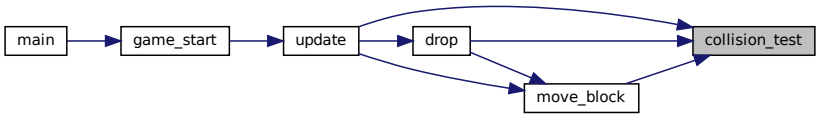
```
int collision_test (
            int command )
```

Definition at line 202 of file update.c.

```
203 {
204     int i, j;
205     int tempx, tempy;
206     int oldx, oldy;
207     int temp_block_state;
208     char(*block_pointer)[4][4][4];
209     char temp_tetris_table[21][10];
210
211     oldx = tempx = x;
212     oldy = tempy = y;
213     temp_block_state = block_state;
214
215     switch (command)
216     {
217     case LEFT:
218         tempx--;
219         break;
220     case RIGHT:
221         tempx++;
222         break;
223     case DOWN:
224         tempy++;
225         break;
226     case ROTATE:
227         temp_block_state++;
228         temp_block_state %= 4;
229         break;
230     }
231
232     switch (block_number)
233     {
```

```
234     case I_BLOCK:
235         block_pointer = &i_block;
236         break;
237     case T_BLOCK:
238         block_pointer = &t_block;
239         break;
240     case S_BLOCK:
241         block_pointer = &s_block;
242         break;
243     case Z_BLOCK:
244         block_pointer = &z_block;
245         break;
246     case L_BLOCK:
247         block_pointer = &l_block;
248         break;
249     case J_BLOCK:
250         block_pointer = &j_block;
251         break;
252     case O_BLOCK:
253         block_pointer = &o_block;
254         break;
255     }
256
257     for (i = 0; i < 21; i++)
258     {
259         for (j = 0; j < 10; j++)
260         {
261             temp_tetris_table[i][j] = tetris_table[i][j];
262         }
263     }
264
265     for (i = 0, oldy = y; i < 4; i++, oldy++)
266     {
267         for (j = 0, oldx = x; j < 4; j++, oldx++)
268         {
269             if (oldx > 0 && oldx < 9 && oldy < 20 && oldy > 0)
270             {
271                 if ((*block_pointer)[block_state][i][j] == 1)
272                     temp_tetris_table[oldy][oldx] = 0;
273             }
274         }
275     }
276
277     for (i = 0; i < 4; i++)
278     {
279         for (j = 0; j < 4; j++)
280         {
281
282             if (temp_tetris_table[tempy + i][tempx + j] == 1 && (*block_pointer)[temp_block_state][i][j]
      == 1)
283                 return 1;
284         }
285     }
286
287     return 0;
288 }
```

Here is the caller graph for this function:




#### 5.19.1.3 drop()

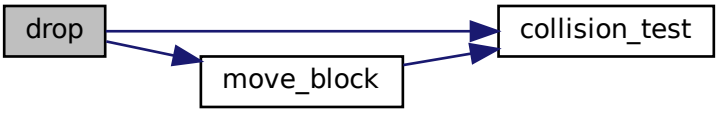
```
int drop (
              void  )
```

Definition at line 291 of file update.c.

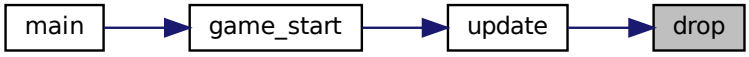
```
292 {
293     while (!collision_test(DOWN))
294         move_block(DOWN);
295
```

```
296     return 0;
297 }
```

Here is the call graph for this function:




Here is the caller graph for this function:




#### 5.19.1.4 move_block()

```
int move_block (
            int command )
```

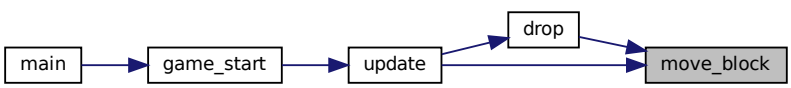
Definition at line 113 of file update.c.

```
114 {
115     int i, j;
116     int newx, newy;
117     int oldx, oldy;
118     int old_block_state;
119     char(*block_pointer)[4][4][4] = NULL;
120 
121     newx = x;
122     newy = y;
123 
124     old_block_state = block_state;
125 
126     if (collision_test(command) == 0)
127     {
128         switch (command)
129         {
130         case LEFT:
131             newx--;
132             break;
133         case RIGHT:
134             newx++;
135             break;
136         case DOWN:
137             newy++;
138             break;
139         case ROTATE:
140             block_state++;
141             block_state %= 4;
142             break;
143         }
144     }
145     else
146     {
147         return 1;
148     }
149 
150     switch (block_number)
```

```
151     {
152     case I_BLOCK:
153         block_pointer = &i_block;
154         break;
155     case T_BLOCK:
156         block_pointer = &t_block;
157         break;
158     case S_BLOCK:
159         block_pointer = &s_block;
160         break;
161     case Z_BLOCK:
162         block_pointer = &z_block;
163         break;
164     case L_BLOCK:
165         block_pointer = &l_block;
166         break;
167     case J_BLOCK:
168         block_pointer = &j_block;
169         break;
170     case O_BLOCK:
171         block_pointer = &o_block;
172         break;
173     }
174 
175     for (i = 0, oldy = y; i < 4; i++, oldy++)
176     {
177         for (j = 0, oldx = x; j < 4; j++, oldx++)
178         {
179             if (oldx > 0 && oldx < 9 && oldy < 20 && oldy > 0)
180                 if ((*block_pointer)[old_block_state][i][j] == 1)
181                     tetris_table[oldy][oldx] = 0;
182         }
183     }
184 
185     x = newx;
186     y = newy;
187 
188     for (i = 0, newy = y; i < 4; i++, newy++)
189     {
190         for (j = 0, newx = x; j < 4; j++, newx++)
191         {
192             if (newx > 0 && newx < 9 && newy < 20 && newy > 0)
193                 if ((*block_pointer)[block_state][i][j] == 1)
194                     tetris_table[newy][newx] = (*block_pointer)[block_state][i][j];
195         }
196     }
197 
198     return 0;
199 }
```

Here is the call graph for this function:


Here is the caller graph for this function:



#### 5.19.1.5 update()

```
int update (
            int signum )
```

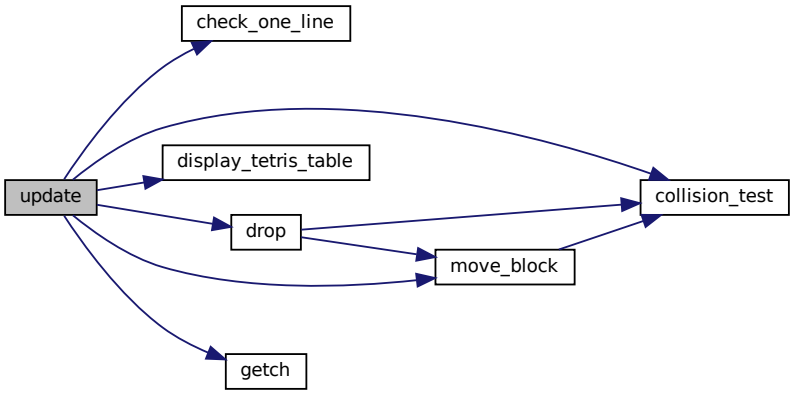
. ,,

Definition at line 6 of file update.c.

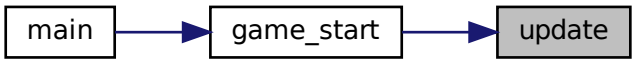
```
7 {
8     static int downcount = 0;
9     static int setcount = 0;
10     static long speedcount = 0;
11     static int countrange = 5;
12     static int firststart = 0;
13
14     char ch;
15
16     srand((unsigned)time(NULL));
17
18     if (firststart == 0)
19     {
20         block_number = rand() % 7;
21         if (firststart == 0)
22             firststart++;
23     }
24
25     display_tetris_table(&countrange);
26     check_one_line();
27
28     if (downcount == countrange - 1)
29     {
30         point += 1;
31         move_block(DOWN);
32     }
33
34     if (speedcount == 499)
35     {
36         if (countrange != 1)
37             countrange--;
38     }
39
40     downcount++;
41     downcount %= countrange;
42     speedcount++;
43     speedcount %= 500;
44
45     if (x == 3 && y == 0)
46     {
47         if (collision_test(LEFT) || collision_test(RIGHT) || collision_test(DOWN) ||
       collision_test(ROTATE))
48         {
49             printf("\n Game End! \n");
50             downcount = 0;
51             setcount = 0;
52             speedcount = 0;
53             countrange = 5;
54             firststart = 0;
55             game = GAME_END;
56         }
57     }
58
59     if (collision_test(DOWN))
60     {
61         if (setcount == 9)
62         {
63             block_number = next_block_number;
64             next_block_number = rand() % 7;
65             block_state = 0;
66             x = 3;
67             y = 0;
68         }
69         setcount++;
70         setcount %= 10;
71     }
72
73     ch = getch();
74
75     switch (ch)
76     {
77     case 74:
78     case 106:
79         move_block(LEFT);
80         break;
81     case 76:
82     case 108:
83         move_block(RIGHT);
84         break;
```

```
85      case 75:
86      case 107:
87          move_block(DOWN);
88          break;
89      case 73:
90      case 105:
91          move_block(ROTATE);
92          break;
93      case 65:
94      case 97:
95          drop();
96          break;
97      case 80:
98      case 112:
99          downcount = 0;
100          setcount = 0;
101          speedcount = 0;
102          countrange = 5;
103          firststart = 0;
104          game = GAME_END;
105          break;
106      default:
107          break;
108      }
109      return 0;
110  }
```

Here is the call graph for this function:




Here is the caller graph for this function:



# Index