Symbol の解説

工藤信一朗

2021 年 7 月 11 日

目次

1 Symbol?

こちらは 2~4 年生を対象とした。C 言語の知識があることが前提とした問題でした。問題の意図としては、プログラムのシンボルという概念とグローバル変数、ローカル変数がどのようになっているかについて触れてもらいたかった。

1.1 ソースコード

1.1.1 local_symbol1.c

```c
/* ベースとなるプログラム */

#include <stdint.h>
#include <stdio.h>

extern char base_char[32];
extern char char_0;
extern char char_0;
extern char char_1;
extern char char_2;
```

```
extern char char_3;
extern char char_4;
extern char char_5;
extern char char_6;
extern char char_7;
extern char char_8;
extern char char_9;
extern char char_10;
extern char char_11;
extern char char_12;
extern char char_13;
extern char char_14;
extern char char_15;
extern char char_16;
extern char char_17;
extern char char_18;
extern char char_19;
extern char char_20;
extern char char_21;
extern char char_22;
extern char char_23;
extern char char_24;
extern char char_25;
extern char char_26;
extern char char_27;
extern char char_28;
extern char char_29;
extern char char_30;


int main(){

    base_char[0] += char_0;
    base_char[1] += char_1;
    base_char[2] += char_2;
    base_char[3] += char_3;
```

```
    base_char[4] += char_4;
    base_char[5] += char_5;
    base_char[6] += char_6;
    base_char[7] += char_7;
    base_char[8] += char_8;
    base_char[9] += char_9;
    base_char[10] += char_10;
    base_char[11] += char_11;
    base_char[12] += char_12;
    base_char[13] += char_13;
    base_char[14] += char_14;
    base_char[15] += char_15;
    base_char[16] += char_16;
    base_char[17] += char_17;
    base_char[18] += char_18;
    base_char[19] += char_19;
    base_char[20] += char_20;
    base_char[21] += char_21;
    base_char[22] += char_22;
    base_char[23] += char_23;
    base_char[24] += char_24;
    base_char[25] += char_25;
    base_char[26] += char_26;
    base_char[27] += char_27;
    base_char[28] += char_28;
    base_char[29] += char_29;
    base_char[30] += char_30;



    printf("htp-ctf{%s}", base_char);
    printf("\n");
    return 0;
}
```

1.1.2 local_symbol2.c

```c
/* こちらのファイルで各種変数の宣言をグローバルで行う */

#include <stdint.h>
#include <stdio.h>              /* test用 */


/* Enjoyable_low_level_programming */
char base_char[32] = {0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
                      0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
                      0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
                      0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, '\0'};

/*w_level_programming */
char char_0 = 4;
char char_1 = 45;
char char_2 = 41;
char char_3 = 46;
char char_4 = 56;
char char_5 = 32;
char char_6 = 33;
char char_7 = 43;
char char_8 = 36;
char char_9 = 30;
char char_10 = 43;
char char_11 = 46;
char char_12 = 54;
char char_13 = 30;
char char_14 = 43;
char char_15 = 36;
char char_16 = 53;
char char_17 = 36;
char char_18 = 43;
char char_19 = 30;
char char_20 = 47;
```

```
char char_21 = 49;
char char_22 = 46;
char char_23 = 38;
char char_24 = 49;
char char_25 = 32;
char char_26 = 44;
char char_27 = 44;
char char_28 = 40;
char char_29 = 45;
char char_30 = 38;
```

1.2 解法

1. まずは実行してみよう
   そのまま実行しようとすると以下のようなエラーを吐く。

```
$ gcc local_symbol1 local_symbol2
/usr/bin/ld: local_symbol1: in function `main':
local_symbol1.c:(.text+0xb): undefined reference to `base_char'
/usr/bin/ld: local_symbol1.c:(.text+0x1c): undefined reference to `base_char'
/usr/bin/ld: local_symbol1.c:(.text+0x23): undefined reference to `base_char'
/usr/bin/ld: local_symbol1.c:(.text+0x34): undefined reference to `base_char'
/usr/bin/ld: local_symbol1.c:(.text+0x3b): undefined reference to `base_char'
/usr/bin/ld: local_symbol1:local_symbol1.c:(.text+0x4c): more undefined refer
ences to `base_char' follow
/usr/bin/ld: local_symbol1: in function `main':
local_symbol1.c:(.text+0x134): undefined reference to `char_12'
/usr/bin/ld: local_symbol1.c:(.text+0x13c): undefined reference to `base_char'
/usr/bin/ld: local_symbol1.c:(.text+0x143): undefined reference to `base_char'
/usr/bin/ld: local_symbol1.c:(.text+0x154): undefined reference to `base_char'
/usr/bin/ld: local_symbol1.c:(.text+0x15b): undefined reference to `base_char'
/usr/bin/ld: local_symbol1.c:(.text+0x16c): undefined reference to `base_char'
/usr/bin/ld: local_symbol1:local_symbol1.c:(.text+0x173): more undefined refer
ences to `base_char' follow
/usr/bin/ld: local_symbol1: in function `main':
local_symbol1.c:(.text+0x194): undefined reference to `char_16'
```

```
/usr/bin/ld: local_symbol1.c:(.text+0x19c): undefined reference to `base_char'
/usr/bin/ld: local_symbol1.c:(.text+0x1a3): undefined reference to `base_char'
/usr/bin/ld: local_symbol1.c:(.text+0x1b4): undefined reference to `base_char'
/usr/bin/ld: local_symbol1.c:(.text+0x1bb): undefined reference to `base_char'
/usr/bin/ld: local_symbol1.c:(.text+0x1cc): undefined reference to `base_char'
/usr/bin/ld: local_symbol1:local_symbol1.c:(.text+0x1d3): more undefined refer
ences to `base_char' follow
/usr/bin/ld: local_symbol1: in function `main':
local_symbol1.c:(.text+0x1f4): undefined reference to `char_20'
/usr/bin/ld: local_symbol1.c:(.text+0x1fc): undefined reference to `base_char'
/usr/bin/ld: local_symbol1.c:(.text+0x203): undefined reference to `base_char'
/usr/bin/ld: local_symbol1.c:(.text+0x214): undefined reference to `base_char'
/usr/bin/ld: local_symbol1.c:(.text+0x21b): undefined reference to `base_char'
/usr/bin/ld: local_symbol1.c:(.text+0x22c): undefined reference to `base_char'
/usr/bin/ld: local_symbol1:local_symbol1.c:(.text+0x233): more undefined refer
ences to `base_char' follow
/usr/bin/ld: local_symbol1: in function `main':
local_symbol1.c:(.text+0x2cc): undefined reference to `char_29'
/usr/bin/ld: local_symbol1.c:(.text+0x2d4): undefined reference to `base_char'
/usr/bin/ld: local_symbol1.c:(.text+0x2db): undefined reference to `base_char'
/usr/bin/ld: local_symbol1.c:(.text+0x2ec): undefined reference to `base_char'
/usr/bin/ld: local_symbol1.c:(.text+0x2f3): undefined reference to `base_char'
collect2: error: ld returned 1 exit status
```

これ見ると、base_char,char_12,char_16,char_20,char_29 が参照できないと表示されています。

2. シンボルがどのような状態か、確認してみましょう
   シンボルの状態を確認する方法はいくつかありますが、ここでは readelf コマンドで参照する仕方を説明します。他には nm コマンドなどがあります。
   man readelf とし/symbil で検索を行うと見ることができますが、–symbols か-s で確認をすることができます。

```
$ readelf --symbols local_symbol2

Symbol table '.symtab' contains 40 entries:
   番号:      値          サイズ タイプ  Bind   Vis      索引名
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 0000000000000000     0 FILE    LOCAL  DEFAULT  ABS local_symbol2.c
```

```
 2: 0000000000000000     0 SECTION LOCAL  DEFAULT    1
 3: 0000000000000000     0 SECTION LOCAL  DEFAULT    2
 4: 0000000000000000     0 SECTION LOCAL  DEFAULT    3
 5: 0000000000000000     0 SECTION LOCAL  DEFAULT    5
 6: 0000000000000000     0 SECTION LOCAL  DEFAULT    6
 7: 0000000000000000     0 SECTION LOCAL  DEFAULT    4
 8: 000000000000002c     1 OBJECT  LOCAL  DEFAULT    2 char_12
 9: 0000000000000030     1 OBJECT  LOCAL  DEFAULT    2 char_16
10: 0000000000000034     1 OBJECT  LOCAL  DEFAULT    2 char_20
11: 0000000000000000    32 OBJECT  LOCAL  DEFAULT    2 base_char
12: 000000000000003d     1 OBJECT  LOCAL  DEFAULT    2 char_29
13: 0000000000000037     1 OBJECT  GLOBAL DEFAULT    2 char_23
14: 0000000000000020     1 OBJECT  GLOBAL DEFAULT    2 char_0
15: 0000000000000021     1 OBJECT  GLOBAL DEFAULT    2 char_1
16: 0000000000000022     1 OBJECT  GLOBAL DEFAULT    2 char_2
17: 0000000000000023     1 OBJECT  GLOBAL DEFAULT    2 char_3
18: 0000000000000024     1 OBJECT  GLOBAL DEFAULT    2 char_4
19: 0000000000000025     1 OBJECT  GLOBAL DEFAULT    2 char_5
20: 0000000000000026     1 OBJECT  GLOBAL DEFAULT    2 char_6
21: 0000000000000027     1 OBJECT  GLOBAL DEFAULT    2 char_7
22: 0000000000000028     1 OBJECT  GLOBAL DEFAULT    2 char_8
23: 0000000000000029     1 OBJECT  GLOBAL DEFAULT    2 char_9
24: 000000000000002a     1 OBJECT  GLOBAL DEFAULT    2 char_10
25: 000000000000002b     1 OBJECT  GLOBAL DEFAULT    2 char_11
26: 000000000000002d     1 OBJECT  GLOBAL DEFAULT    2 char_13
27: 000000000000002e     1 OBJECT  GLOBAL DEFAULT    2 char_14
28: 000000000000002f     1 OBJECT  GLOBAL DEFAULT    2 char_15
29: 0000000000000031     1 OBJECT  GLOBAL DEFAULT    2 char_17
30: 0000000000000032     1 OBJECT  GLOBAL DEFAULT    2 char_18
31: 0000000000000033     1 OBJECT  GLOBAL DEFAULT    2 char_19
32: 0000000000000035     1 OBJECT  GLOBAL DEFAULT    2 char_21
33: 0000000000000036     1 OBJECT  GLOBAL DEFAULT    2 char_22
34: 0000000000000038     1 OBJECT  GLOBAL DEFAULT    2 char_24
35: 0000000000000039     1 OBJECT  GLOBAL DEFAULT    2 char_25
36: 000000000000003a     1 OBJECT  GLOBAL DEFAULT    2 char_26
37: 000000000000003b     1 OBJECT  GLOBAL DEFAULT    2 char_27
38: 000000000000003c     1 OBJECT  GLOBAL DEFAULT    2 char_28
```

```
    39: 000000000000003e     1 OBJECT  GLOBAL DEFAULT    2 char_30
```

ここで注目するべきは以下です。

```
     8: 000000000000002c     1 OBJECT  LOCAL  DEFAULT    2 char_12
     9: 0000000000000030     1 OBJECT  LOCAL  DEFAULT    2 char_16
    10: 0000000000000034     1 OBJECT  LOCAL  DEFAULT    2 char_20
    11: 0000000000000000    32 OBJECT  LOCAL  DEFAULT    2 base_char
    12: 000000000000003d     1 OBJECT  LOCAL  DEFAULT    2 char_29
```

ローカルになっている部分があることが解ると思います。これによって、local_symbol1 から参照できていないのです。

3. ローカルになっているシンボルをグローバルに変更する
   シンボルを変更する手法として objcopy コマンドを利用することができます。その中でも–globalize-symbol= というオプションを使用します。

```
objcopy local_symbol2.o --globalize-symbol=char_12
 ↪  --globalize-symbol=char_16 --globalize-symbol=char_20
 ↪  --globalize-symbol=char_29 --globalize-symbol=base_char
```

ここでシンボルを見てみましょう。

```
$ readelf --symbols local_symbol2.o

Symbol table '.symtab' contains 40 entries:
   番号:      値          サイズ タイプ  Bind   Vis      索引名
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 0000000000000000     0 FILE    LOCAL  DEFAULT  ABS local_symbol2.c
     2: 0000000000000000     0 SECTION LOCAL  DEFAULT    1
     3: 0000000000000000     0 SECTION LOCAL  DEFAULT    2
     4: 0000000000000000     0 SECTION LOCAL  DEFAULT    3
     5: 0000000000000000     0 SECTION LOCAL  DEFAULT    5
     6: 0000000000000000     0 SECTION LOCAL  DEFAULT    6
     7: 0000000000000000     0 SECTION LOCAL  DEFAULT    4
     8: 0000000000000030     1 OBJECT  GLOBAL DEFAULT    2 char_16
     9: 0000000000000034     1 OBJECT  GLOBAL DEFAULT    2 char_20
    10: 0000000000000000    32 OBJECT  GLOBAL DEFAULT    2 base_char
    11: 000000000000003d     1 OBJECT  GLOBAL DEFAULT    2 char_29
    12: 000000000000002c     1 OBJECT  GLOBAL DEFAULT    2 char_12
```

```
13: 0000000000000037     1 OBJECT  GLOBAL DEFAULT    2 char_23
14: 0000000000000020     1 OBJECT  GLOBAL DEFAULT    2 char_0
15: 0000000000000021     1 OBJECT  GLOBAL DEFAULT    2 char_1
16: 0000000000000022     1 OBJECT  GLOBAL DEFAULT    2 char_2
17: 0000000000000023     1 OBJECT  GLOBAL DEFAULT    2 char_3
18: 0000000000000024     1 OBJECT  GLOBAL DEFAULT    2 char_4
19: 0000000000000025     1 OBJECT  GLOBAL DEFAULT    2 char_5
20: 0000000000000026     1 OBJECT  GLOBAL DEFAULT    2 char_6
21: 0000000000000027     1 OBJECT  GLOBAL DEFAULT    2 char_7
22: 0000000000000028     1 OBJECT  GLOBAL DEFAULT    2 char_8
23: 0000000000000029     1 OBJECT  GLOBAL DEFAULT    2 char_9
24: 000000000000002a     1 OBJECT  GLOBAL DEFAULT    2 char_10
25: 000000000000002b     1 OBJECT  GLOBAL DEFAULT    2 char_11
26: 000000000000002d     1 OBJECT  GLOBAL DEFAULT    2 char_13
27: 000000000000002e     1 OBJECT  GLOBAL DEFAULT    2 char_14
28: 000000000000002f     1 OBJECT  GLOBAL DEFAULT    2 char_15
29: 0000000000000031     1 OBJECT  GLOBAL DEFAULT    2 char_17
30: 0000000000000032     1 OBJECT  GLOBAL DEFAULT    2 char_18
31: 0000000000000033     1 OBJECT  GLOBAL DEFAULT    2 char_19
32: 0000000000000035     1 OBJECT  GLOBAL DEFAULT    2 char_21
33: 0000000000000036     1 OBJECT  GLOBAL DEFAULT    2 char_22
34: 0000000000000038     1 OBJECT  GLOBAL DEFAULT    2 char_24
35: 0000000000000039     1 OBJECT  GLOBAL DEFAULT    2 char_25
36: 000000000000003a     1 OBJECT  GLOBAL DEFAULT    2 char_26
37: 000000000000003b     1 OBJECT  GLOBAL DEFAULT    2 char_27
38: 000000000000003c     1 OBJECT  GLOBAL DEFAULT    2 char_28
39: 000000000000003e     1 OBJECT  GLOBAL DEFAULT    2 char_30
```

すべて、グローバル変数に変換されていることがわかります。
この後、通常通りコンパイルすることで a.out ファイルが生成され、実行することでフラグがわかります。

```
gcc local_symbol1.o local_symbol2.o
```

1.3 フラグ

htp-ctf{Enjoyable_low_level_programming}

symbol?

問題の意図

この問題は2~4年生を対象としていました。シンボルはオブジェクトファイルにした後でも参照先の制限を変更できることを知って欲しかった。

解法

objcopy(man objcopy)コマンドの--globalize-symbol=<symbol>オプションを利用し、シンボルの参照を変更する。

解説

C言語では、基本ファイル単位での隠蔽が行われる。詳しく言うと、シンボルという関数や変数に割り当てられている名前がある。