Zad 3.

piątek, 5 maja 2023 16:45

Zadanie 3. Rozważmy program składający się z dwóch plików źródłowych:

```c
/* mismatch-a.c */
void p2(void);

int main(void) {
  p2();
  return 0;
}
```

```c
/* mismatch-b.c */
#include <stdio.h>

char main;

void p2(void) {
  printf("0x%x\n", main);
}
```

symbol silny

symbol słaby

referencja do

Po uruchomieniu program drukuje pewien ciąg znaków i kończy działanie bez zgłoszenia błędu. Czemu tak się dzieje? Skąd pochodzi wydrukowana wartość? Czym różni się **symbol silny** od **słabego**? Zauważ, że zmienna «main» w pliku «mismatch-b.c» jest niezainicjowana. Co by się stało, gdybyśmy w funkcji «p2» (1) przypisali wartość pod zmienną «main»? Co by się zmieniło gdybyśmy w pliku «mismatch-b.c» zainicjowali (2) zmienną «main» w miejscu jej definicji? Czemu dobrym pomysłem jest przekazywanie opcji «-fno-common» do kompilatora?

program drukuje 0x48

(2) byłyby wtedy dwa silne symbole i konsolidator by się obraził

(1) segfault w trakcie run-time'u

-fno-common zapobiega błędom związanym z kilkoma definicjami jednego symbolu, informując o nich programiste

```
mismatch-a.o:     file format elf64-x86-64


Disassembly of section .text:

0000000000000000 <main>:
   0:   48 83 ec 08             sub    $0x8,%rsp
   4:   e8 00 00 00 00          call   9 <main+0x9>
   9:   b8 00 00 00 00          mov    $0x0,%eax
   e:   48 83 c4 08             add    $0x8,%rsp
  12:   c3                      ret
```

to się drukuje

```
mismatch-b.o:     file format elf64-x86-64


Disassembly of section .text:

0000000000000000 <p2>:
   0:   f3 0f 1e fa             endbr64
   4:   48 83 ec 08             sub    $0x8,%rsp
   8:   0f be 15 00 00 00 00    movsbl 0x0(%rip),%edx        # f <p2+0xf>
   f:   be 00 00 00 00          mov    $0x0,%esi
  14:   bf 01 00 00 00          mov    $0x1,%edi
  19:   b8 00 00 00 00          mov    $0x0,%eax
  1e:   e8 00 00 00 00          call   23 <p2+0x23>
  23:   48 83 c4 08             add    $0x8,%rsp
  27:   c3                      ret
```