Architektury systemów komputerowych 2016

Lista zadań nr 6

Na zajęcia 6, 7, 11 i 12 kwietnia 2016

UWAGA! Rozwiązywanie zadań z tej listy będzie wymagało dostępu do komputera z systemem Linux dla platformy x86-64. Prowadzący zakłada, że zainstalowana dystrybucja będzie bazowała na Debianie (np. Ubuntu).

Zadanie 1. Poniżej podano zawartość pliku `swap.c`:

```
1  extern int buf[];
2
3  int *bufp0 = &buf[0];
4  static int *bufp1;
5
6  static void incr() {
7    static int count = 0;
8    count++;
9  }
10
11 void swap() {
12   int temp;
13   incr();
14   bufp1 = &buf[1];
15   temp = *bufp0;
16   *bufp0 = *bufp1;
17   *bufp1 = temp;
18 }
```

Dla każdego symbolu zdefiniowanego lub wykorzystywanego w `swap.o` podaj:

- czy będzie on wymieniony w tablicy symboli `.symtab` modułu `swap.o`?
- typ symbolu (`local`, `global`, `extern`),
- sekcję, do której się odnosi (`.text`, `.data`, `.bss`).

Zadanie 2. Rozważmy poniższe dwa pliki źródłowe.

```
1 /* foo.c */
2 void p2(void);
3
4 int main() {
5   p2();
6   return 0;
7 }
```

```
1 /* bar.c */
2 #include <stdio.h>
3
4 char main;
5
6 void p2() {
7   printf("0x%x\n", main);
8 }
```

Po skompilowaniu i uruchomieniu na platformie Linux x86-64 program drukuje ciąg znaków `0x48\n` i kończy działanie bez zgłoszenia błędu. Zauważ, że zmienna `main` z pliku `bar.c` nie jest zainicjowana. Czemu tak się dzieje i skąd pochodzi wartość `0x48`?

Zadanie 3. Poniższy kod w języku `C` skompilowano z opcją `-Og`, a następnie poddano deasemblacji sekcję `.text` otrzymanej jednostkę translacji. Tablicę skoków dla instrukcji wyboru kompilator umieścił w sekcji `.rodata` i wypełnił zerami.

Dla obydwu sekcji określ pod jakimi miejscami znajdują się relokacje, a następnie podaj zawartość tablicy relokacji `.rela.text` i `.rela.data`, tj. listę rekordów składających się z:

- przesunięcia relokacji względem początku sekcji,
- typu relokacji,
- nazwy symbolu.

```
1  int relo3(int val) {
2    switch (val) {
3      case 100:
4        return(val);
5      case 101:
6        return(val+1);
7      case 103: case 104:
8        return(val+3);
9      case 105:
10       return(val+5);
11     default:
12       return(val+6);
13   }
14 }
```

```
0000000000000000 <relo3>:
 0: 8d 47 9c              lea    -0x64(%rdi),%eax
 3: 83 f8 05              cmp    $0x5,%eax
 6: 77 15                 ja     1d <relo3+0x1d>
 8: 89 c0                 mov    %eax,%eax
 a: ff 24 c5 00 00 00 00  jmpq   *0x0(,%rax,8)
11: 8d 47 01              lea    0x1(%rdi),%eax
14: c3                    retq
15: 8d 47 03              lea    0x3(%rdi),%eax
18: c3                    retq
19: 8d 47 05              lea    0x5(%rdi),%eax
1c: c3                    retq
1d: 8d 47 06              lea    0x6(%rdi),%eax
20: c3                    retq
21: 89 f8                 mov    %edi,%eax
23: c3                    retq
```

Zadanie 4. Zapoznaj się z narzędziami do analizy plików `ELF` i bibliotek statycznych, tj. `objdump`, `readelf` i `ar`; a następnie odpowiedz na następujące pytania:

1. Ile modułów translacji zawierają biblioteki `libc.a` i `libm.a`? (katalog `/usr/lib/x86_64-linux-gnu`)
2. Czy polecenie `gcc -Og` generuje inny kod wykonywalny niż `gcc -Og -g`?
3. Z jakich bibliotek współdzielonych korzysta interpreter języka `Python`? (plik `/usr/bin/python`)

Zaprezentuj w jaki sposób można dojść do odpowiedzi korzystając z w/w poleceń.

Zadanie 5. Na podstawie rozdziału 7.12 podręcznika *„Computer Systems: A Programmer's Perspective"* wytłumacz przystępnie na czym polega proces dynamicznej konsolidacji. Jaką rolę w tym procesie odgrywają sekcje `GOT` i `PLT`? Zauważ, że symbole są **wiązane leniwie** — co to znaczy?

Zadanie 6. Język `C++` pozwala na przeciążanie funkcji, tj. dopuszcza stosowanie wielu funkcji o tej samej nazwie, ale różnej sygnaturze. Jak już wiemy, symbole, na których operuje konsolidator, są beztypowe. Powstaje zatem problem unikalnej reprezentacji symboli funkcji przeciążonych. Wytłumacz na czym polega **dekorowanie nazw** (ang. *name mangling*)?

Przy pomocy narzędzia `c++filt` przekształć poniższe symbole konsolidatora do sygnatur funkcji języka `C++`, a następnie omów znaczenie poszczególnych fragmentów symbolu.

1. `_Z4funcPKcRi`
2. `_ZN3Bar3bazEPc`
3. `_ZN3BarC1ERKS_`
4. `_ZN3foo6strlenER6string`

Zadanie 7. Odpowiedz na podstawie dokumentu GNU as: Assembler Directives jakiej dyrektywy należy użyć z poziomu asemblera, żeby:

1. zadeklarować dany symbol jako globalny / lokalny / słaby?
2. utworzyć sekcję danych tylko do odczytu?
3. określić, że symbol odnosi się do funkcji lub zmiennej?
4. zarezerwować `n` bajtów na zmienną niezainicjowaną `var`?