

Architektury systemów komputerowych

Lista zadań nr 8

Na zajęcia 29 kwietnia 2021

Zadania z tej listy należy rozwiązywać na komputerze z systemem operacyjnym LINUX dla platformy x86-64. Prowadzący zakłada, że zainstalowana dystrybucja będzie bazowała na DEBIAN-ie (np. UBUNTU lub MINT).

Do poniższej listy załączono na stronie przedmiotu pliki źródłowe wraz z plikiem Makefile.

UWAGA! W trakcie prezentacji rozwiązań należy zdefiniować i wyjaśnić pojęcia, które zostały oznaczone **wytluszczoną** czcionką.

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 void swap() {
11     int temp;
12     incr();
13     bufp1 = &buf[1];
14     temp = *bufp0;
15     *bufp0 = *bufp1;
16     *bufp1 = temp;
17 }
```

Wygeneruj **plik relokowalny** «swap.o», po czym na podstawie wydruku polecenia «readelf -t -s» dla każdego elementu tablicy symboli podaj:

- adres symbolu względem początku sekcji,
- typ symbolu – tj. «local», «global», «extern»,
- rozmiar danych, na które wskazuje symbol,
- numer i nazwę sekcji – tj. «.text», «.data», «.bss» – do której odnosi się symbol.

Co przechowują sekcje «.strtab» i «.shstrtab»?

Zadanie 2. Rozważmy program składający się z dwóch plików źródłowych:

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

Po uruchomieniu program drukuje pewien ciąg znaków i kończy działanie bez zgłoszenia błędu. Cemu tak się dzieje? Skąd pochodzi wydrukowana wartość? Zauważ, że zmienna «main» w pliku «mismatch-a.c» jest niezainicjowana. Co by się stało, gdybyśmy w funkcji «p2» przypisali wartość pod zmienną «main»? Co by się zmieniło gdybyśmy w pliku «mismatch-b.c» zainicjowali zmienną «main» w miejscu jej definicji?

Zadanie 3. Zapoznaj się z narzędziami do analizy **plików relokowalnych** w formacie ELF i bibliotek statycznych, tj. «objdump», «readelf» i «ar»; a następnie odpowiedz na następujące pytania:

1. Ile plików zawierają biblioteki «libc.a» i «libm.a» (katalog «/usr/lib/x86_64-linux-gnu»)?
2. Czy wywołanie kompilatora z opcją «-Og» generuje inny kod wykonywalny niż «-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 podanych poleceń.

Zadanie 4. Rozważmy program składający się z dwóch plików źródłowych:

```
1 /* str-a.c */           1 /* str-b.c */
2 #include <stdio.h>       2 char *sometr(void) {
3                           3     return "Hello, world!";
4 char *sometr(void);     4 }
5
6 int main(void) {
7     char *s = sometr();
8     s[5] = '\0';
9     puts(s);
10    return 0;
11 }
```

Po uruchomieniu program kończy się z błędem dostępu do pamięci. Przy pomocy debuggera gdb zatrzymaj się w miejscu wystąpienia awarii i wyjaśnij jej przyczynę. Gdzie została umieszczona stała znakowa "Hello, world! "? Popraw ten program tak, by się poprawnie zakończył. Gdzie został umieszczony ciąg znaków po poprawce? Nie wolno modyfikować sygnatury procedury «sometr» i pliku «str-a.c», ani korzystać z dodatkowych procedur.

Zadanie 5. Posiłkując się narzędziem «objdump» podaj rozmiary sekcji «.data» i «.bss» plików «bar.o» i «foo.o». Wskaż rozmiar i pozycje symboli względem początków odpowiednich sekcji. Wyjaśnij znaczenie opcji «-fno-common» przekazywanej do kompilatora.

```
1 /* bar.c */           1 /* foo.c */
2 int bar = 42;          2 long foo = 19;
3 short dead[15];        3 char code[17];
```

Na czym polega **częściowa konsolidacja** z użyciem opcji «-r» do polecenia «ld»? Czym różni się sposób wygenerowania plików «merge-1.o» i «merge-2.o»? Na podstawie **mapy konsolidacji** porównaj pozycje symboli i rozmiary sekcji w plikach wynikowych. Z czego wynikają różnice skoro konsolidator nie dysponuje informacjami o typach języka C?

Zadanie 6. Plik **wykonywalny** powstały w wyniku kompilacji poniższych plików źródłowych powinien być nie dłuższy niż 1KiB. Na podstawie **nagłówka pliku ELF** wskaż w zdeasemblowanym pierwszą instrukcję, którą wykona procesor po wejściu do programu. Na podstawie **nagłówków programu** wskaż pod jaki adres wirtualny zostanie załadowany **segment** z sekcją «.text».

```
1 /* start.c */           1 /* even.c */           1 /* odd.c */
2 int is_even(long);      2 int is_odd(long n);     2 int is_even(long n);
3                           3                           3
4 void _start(void) {     4 int is_even(long n) {   4 int is_odd(long n) {
5     asm volatile(       5     if (n == 0)         5     if (n == 0)
6         "syscall"       6         return 1;      6         return 0;
7         : /* no output */ 7     else               7     else
8         : "a" (0x3c /* exit */), 8     return is_odd(n - 1); 8     return is_even(n - 1);
9         "D" (is_even(42))); 9 }                       9 }
10 }
```

Zapoznaj się ze **skrypcem konsolidatora** w pliku «main.lds». Na podstawie **dokumentacji**¹ wyjaśnij jak skrypt kieruje procesem konsolidacji poszczególnych sekcji i tworzeniem nagłówków programu.

Zadanie 7. Na podstawie [1, §7.7.1] wytłumacz uczestnikom zajęć składowe **rekordów relokacji** przechowywanych w sekcjach «.rel.text» i «.rel.data». Czy możliwe jest by assembler utworzył sekcję «.rel.bss»? Czym się różnią relokacje typu «R_X86_64_PC32» od «R_X86_64_32»? Na podstawie [1, §7.7.2] podręcznika zreferuj **algorytm relokowania** odwołań do symboli.

Wskazówka: Pozostałe typy relokacji zostały opisane w [2, §4.4.1].

Zadanie 8. Na podstawie wydruku polecenia «objdump -r -d» wskaż w kodzie źródłowym z zadania 6 miejsca występowania odwołań do symboli, które wymagają relokacji. Następnie pokaż jak odwołania zostały uzupełnione w pliku wykonywalnym. Wydrukuj tabele rekordów relokacji poszczególnych plików konsolidowanych, a także fragment mapy konsolidacji opisujący złączoną sekcję «.text». Następnie oblicz wartości, które należy wpisać w miejsce odwołania do symbolu, zgodnie z algorytmem z poprzedniego zadania.

¹<https://sourceware.org/binutils/docs/ld/Scripts.html>

Literatura

- [1] „Computer Systems: A Programmer's Perspective”
Randal E. Bryant, David R. O'Hallaron; Pearson; 3rd edition, 2016
- [2] „System V Application Binary Interface AMD64 Architecture Processor Supplement”
H.J. Lu, Michael Matz, Milind Girkar, Jan Hubička, Andreas Jaeger, Mark Mitchell;
<https://github.com/hjl-tools/x86-psABI/wiki/x86-64-psABI-1.0.pdf>