Architektury systemów komputerowych

Lista zadań nr 8

Na zajęcia 30 kwietnia 2020

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 **wytłuszczoną** 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. Czemu 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 «`-O1 -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:

```c
/* str-a.c */
#include <stdio.h>

char *somestr(void);

int main(void) {
  char *s = somestr();
  s[5] = '\0';
  puts(s);
  return 0;
}
```

```c
/* str-b.c */
char *somestr(void) {
  return "Hello, world!";
}
```

Po uruchomieniu program kończy się z błędem dostępu do pamięci. Dlaczego? 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 «`somestr`» 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.

```c
/* bar.c */
int bar = 42;
short dead[15];
```

```c
/* foo.c */
long foo = 19;
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`».

```c
/* start.c */
int is_even(long);

void _start(void) {
  asm volatile(
    "syscall"
    : /* no output */
    : "a" (0x3c /* exit */),
      "D" (is_even(42)));
}
```

```c
/* even.c */
int is_odd(long n);

int is_even(long n) {
  if (n == 0)
    return 1;
  else
    return is_odd(n - 1);
}
```

```c
/* odd.c */
int is_even(long n);

int is_odd(long n) {
  if (n == 0)
    return 0;
  else
    return is_even(n - 1);
}
```

Zapoznaj się ze **skryptem 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] opowiedz jakie dane przechowują sekcje «`.rel.text`» i «`.rel.data`». Czy możliwe jest by asembler 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 relokacji.

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 **relokacji**. Następnie pokaż jak relokacje zostały wypełnione w pliku wykonywalnym. Wydrukuj tabele relokacji poszczególnych plików konsolidowalnych, a także fragment mapy konsolidacji opisujący złączoną sekcję «`.text`». Następnie oblicz wartości, które należy wpisać w miejsce relokacji, 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