Systemy komputerowe

Lista zadań nr 6

Na zajęcia 20 – 23 kwietnia 2020

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

```
1  #include "stdio.h"
2
3  static int global = 15210;
4
5  static void set_global(int val) {
6          global = val;
7  }
```

```
10 int main(void) {
11         printf("before: %d\n", global);
12         set_global(15213);
13         printf("after: %d\n", global);
14         return 0;
15 }
```

Polecenie `gcc main.c -o main` jest równoważne ciągowi poleceń
`cpp -o main_p.c main.c; gcc -S main_p.c; as -o main.o main_p.s; gcc -o main main.o`.

- Jaka jest rola poszczególnych poleceń w tym ciągu?
- Skąd pochodzi kod, który znalazł się w pliku `main_p.c`?
- Co zawiera plik `main_p.s`. Zauważ etykiety odpowiadające zmiennej `global` i obydwu funkcjom. W jaki sposób przyporządkować etykiecie jej typ?
- Poleceniem `objdump -t` wyświetl tablicę symboli pliku `main.o`. Jakie położenie wg. tej tablicy mają symbole `global` i `set_global`?
- Poleceniem `objdump -h` wyświetl informacje o sekcjach w pliku `main.o`. Dlaczego adres sekcji `.text` i `.data` to 0? Jakie są adresy tych sekcji w pliku wykonywalnym `main`?

Zadanie 2. Poniżej podano zawartość pliku `swap.c`:

```
1  extern int buf[];
2
3  int *bufp0 = &buf[0];
4  static int *bufp1;
5  int intvalue = 0x77;
6
7  static void incr() {
8    static int count = 0;
9    count++;
10 }
```

```
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 3. Rozważmy program skompilowany z opcją `-Og` składający się z dwóch plików źródłowych:

```c
/* oof.c */
void p2(void);

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

```c
/* rab.c */
#include <stdio.h>

char main;

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

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 `rab.c` jest niezainicjowana. Co by się stało, gdybyśmy w funkcji `p2` przypisali wartość pod zmienną `main`? Co by się zmieniło gdybyśmy w pliku `rab.c` zainicjowali zmienną `main` w miejscu jej definicji? Odpowiedzi uzasadnij posługując się narzędziem `objdump`.

Wskazówka: Może się przydać opcja -d polecenia objdump

Zadanie 4. Które wiersze w kodzie z zadania drugiego będą wymagać dodania wpisu do tablicy relokacji?

Wskazówka: Zastanów się jakie dodatkowe informacje należy umieścić w plikach relokowalnych, by umieć powiązać miejsca wywołania procedur z położeniem procedury w skonsolidowanym pliku wykonywalnym. Mogą przydać się opcje -d oraz -d -r narzędzia objdump.

Zadanie 5. Wpis w tablicy symboli dla zmiennej `buf` w pliku `swap.o` wskazuje, że zmienna ta znajduje się w sekcji `UNDEF`. Zmodyfikuj deklarację zmiennej `buf` w pliku `swap.c` tak, by wpis w tablicy symboli dla `buf` wskazywał na `COMMON`. Jaka jest rola tych wpisów w procesie tworzenia pliku wykonywalnego przez linker?

Zadanie 6. Dla podanych poniżej plików źródłowych wygeneruj pliki relokowalne przekazując kompilatorowi opcję «`-fno-common`». Następnie wykonaj konsolidację na dwa sposoby:

```
ld -M=merge-1.map -r -o merge-1.o foo.o bar.o
ld -M=merge-2.map -r -o merge-2.o bar.o foo.o
```

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.

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

```c
/* foo.c */
long foo = 19;
char code[17];
```

Wyjaśnij znaczenie opcji «`-fno-common`» przekazywanej do kompilatora. 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** (pliki z rozszerzeniem «`.map`») porównaj pozycje symboli i rozmiary sekcji w plikach wynikowych. Z czego wynikają różnice skoro konsolidator nie dysponuje informacjami o typach języka C?

Zadanie 7 (2pkty). Skompiluj poniższe pliki poleceniami:

```
gcc -fno-pie -c even.c odd.c start.c
ld -static -M=main.map even.o odd.o start.o -o main
```

Na podstawie **nagłówka pliku ELF** wskaż w zdeasemblowanym pliku 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);
}
```

Wskaż w kodzie źródłowym miejsca występowania **relokacji**. Zidentyfikuj je w wydruku polecenia «`objdump -r -d`», po czym pokaż jak zostały wypełnione w pliku wykonywalnym. Na podstawie rozdziału §7.7 podręcznika „Computer Systems: A Programmer's Perspective" zreferuj algorytm relokacji. Wydrukuj tabele relokacji plików relokowalnych, fragment mapy konsolidacji opisujący złączoną sekcję «`.text`», po czym oblicz ręcznie wartości, które należy wpisać w miejsce relokacji.

Zadanie 8. Używając narzędzi do analizy **plików relokowalnych** w formacie `ELF` i bibliotek statycznych, tj. `objdump`, `readelf` i `ar` odpowiedz na następujące pytania:

1. Ile plików 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`)?