PB173 – Binární programování Linux

III. Binární objektové soubory

Jiri Slaby

ITI, Fakulta informatiky

1. 10. 2013

Objektové soubory

- Vznikají při překladu i linkování
  - Výstup překladače (`gcc -c`)
  - Výstup linkeru jako:
    - Dynamická knihovna
    - Spustitelná binárka
- Literatura
  - System V Application Binary Interface (ELF)

Obsah objektových souborů

- Obvykle
  - Informace o souboru
    - Cílová architektura, struktura, . . .
  - Kód
  - Data
  - Ostatní informace potřebné k běhu/linkování
    - Relokace, informace o symbolech atd.
- Ale také např.
  - Ladicí informace
  - Informace o překladači
- Tyto informace v oddílech (sekcích)

Formáty objektových souborů

- Několik forem/standardů
  - a.out (UNIX/starý Linux)
  - *ELF* (Linux)
  - COM (DOS, čistě kód+data, nic víc)
  - COFF (UNIX, ELF ho převálcoval)
  - MZ (DOS .exe)
  - PE (Windows .exe, vychází z COFF)

Demo: DosBox a reboot přes COM (`ea f0 ff 00 f0`)

Úkol

Práce s PE

1. Stáhněte si nějakou Windows aplikaci
   - Cokoliv, co je .exe (PE)
2. Vypište si informace společné formátům (`objdump -f`)
3. Vypište si informace závislé na formátu (`objdump -p`)
   - Zjistěte DLL, na kterých váš program závisí
4. Porovnejte sekce s nějakým ELFem (`objdump -h`)
   - Jakákoliv binárka z `/usr/bin`

ELF

Executable and Linkable Format

- Hlavní souborový formát na Linuxu
- Přenositelný, relokovatelný, rozšiřitelný
- Struktura
  - ELF hlavička
  - Hlavičky programové
    - Při spouštění
  - Hlavičky sekcí
    - Při linkování, ladění apod.
  - Sekce
    - Odkazovány z obou typů hlaviček
- `readelf` je speciální `objdump` pro ELF

ELF hlavička

- `readelf -h`
- Magické číslo (`0x7f 'E''L''F'`)
- Cílová architektura a stroj
- Počty hlaviček

Úkol: výpis ELF hlavičky a zjištění počtu obou typů hlaviček

Sekce

- `readelf -S` (obsah sekce: `readelf -x`)
- Vytvářené překladačem/linkerem
- Čtené překladačem/linkerem/interpretrem
- Předdefinované sekce:
  - `.text`: kód
  - `.data`: nekonstantní data (proměnné)
  - `.rodata`: konstantní data (řetězce apod.)
  - `.bss`: data, která se inicializují na 0
  - `.interp`: interpretr
  - `.symtab`: tabulka symbolů pro ladění (`strip`)
  - `.dynsym`: tabulka symbolů pro dynamický linker
  - `.gnu_debuglink`: odkaz na soubor s ladicími informacemi

Demo: ukázka sekcí

Úkol

Práce se sekcemi

1. Vytvořte si soubor s `puts("hello")` a `puts("world")`
2. Vytvořte si vlastní sekci s daty

   ```
   int __attribute__ ((section(".data.my_section"))) x = 3;
   ```

3. Vytvořte si vlastní sekci s funkcí (`.text.my_section`)
4. Přeložte do `.o` a poté i slinkujte
5. Vypište si seznam sekcí v obou souborech (`readelf -S`)
6. Vypište si obsah sekcí (`readelf -x`)
   - `.rodata` (zkuste i `readelf -p`)
   - V `.o`: `.data.my_section`
   - V `.o`: `.text.my_section` (zkuste i `objdump -d`)

Pozn.: tyto soubory nezahazujte.

libbfd

- Knihovna pro práci s různými binárními formáty (`bfd.h`)
  - a.out, ELF, PE, binární, ...
  - Seznam: `const char **bfd_target_list()`
- A různými architekturami
  - Seznam: `const char **bfd_arch_list()`
- Dokumentace
  - `https://sourceware.org/binutils/docs/bfd/`
  - *Některá užitečná makra bez dokumentace* (jen v `bfd.h`)
- Knihovny při překladu: `gcc ... -lbfd -liberty -ldl -lz`
- První je třeba volat `void bfd_init()`

libbfd: chyby

Jak zjistit, co se stalo?

- Poslední chyba: `bfd_error_type bfd_get_error()`
- Převod na text:
  `const char *bfd_errmsg(bfd_error_type error_tag)`

Typicky:

```
if (!bfd_function (...) )
  errx(1, "bfd_function: %s", bfd_errmsg(bfd_get_error()));
```

Úkol

Vypište všechny formáty podporované vaší `libbfd`

1. Nezapomeňte `bfd_init`
2. Zavolejte `bfd_target_list`
3. Vypište v cyklu
4. Zavolejte `bfd_arch_list`
5. Vypište v cyklu
6. Ověřujte návratové hodnoty podle manuálu
   - Vypisujte chyby při neúspěchu
7. Přeložte a spusťte

libbfd: soubory

- Otevření
  - `bfd *bfd_openr(const char *file, const char *target)`
  - `bfd *bfd_openw(const char *file, const char *target)`
  - `target` je `NULL`
- Ověření
  - *Nutné před prací se souborem*
  - `bfd_boolean bfd_check_format(bfd *abfd, bfd_format format)`
  - `format` je `bfd_object`
- Zavření
  - `bfd_boolean bfd_close(bfd *abfd)`

Bez ověření chyb:

```
abfd = bfd_openr( file ,  NULL);
bfd_check_format(abfd, bfd_object) ;
bfd_close(abfd) ;
```

Úkol

Doplňte otevírání souboru zadaného jako parametr

1. Zavolejte `bfd_openr`
2. Zavolejte `bfd_check_format`
3. Zavolejte `bfd_close`
4. Ověřujte návratové hodnoty podle manuálu
5. Přeložte a spusťte

`libbfd`: sekce

- Iterace: `bfd_map_over_sections`
  - Velikost: `bfd_get_section_size`
  - Obsah: `bfd_get_section_contents`
- Vytvoření: `bfd_make_section_with_flags`
  - Nastavení velikosti: `bfd_set_section_size`
  - Nastavení obsahu: `bfd_set_section_contents`

Úkol

Hexdump sekcí

1. Iterujte přes sekce (`bfd_map_over_sections`)
2. Vypište údaje o každé sekci
   - Název
   - Velikosti
   - Flagy
   - Hexdump prvních 16 bytů obsahu
3. Přeložte
4. Spusťte
   - Pro ELF
   - Pro PE