PB173 – Binární programování Linux

IV. ELF I.

Jiri Slaby

Fakulta informatiky
Masarykova univerzita

15. 10. 2015

ELF

Executable and Linkable Format

- Hlavní souborový formát na Linuxu
- Přenositelný, relokovatelný, rozšiřitelný
- Podpora pro ladicí informace
  - Příště
- `readelf` je specializovaný `objdump` pro ELF
- `libelf` je specializovaná `libbfd` pro ELF
- Literatura
  - System V Application Binary Interface

- `libbfd` je nezávislá na binárním formátu
  - Ale neumí vše, co podporuje ELF
- `libelf` umí jen ELF
  - Ale zvládá téměř všechno, co ELF podporuje
  - Pracuje na nižší úrovni
- `libebl` je nadstavba `libelf`
  - Nemá moc uživatelů

libelf

Detaily

- Knihovna pro práci s ELF
- Nabízí dvě API
  - ELF (`libelf.h`)
  - GELF – více abstraktní, generické (`gelf.h`)
  - Lze použít současně a míchat
- Dokumentace
  - Pro BSD port: A tutorial introduction to libelf
  - V hlavičkových souborech
- Knihovny při překladu: `gcc ... -lelf`
- První je třeba volat `elf_version(EV_CURRENT)`
  - Nesmí být `EV_NONE`

Jak zjistit, co se stalo?

- Poslední chyba: `int elf_errno()`
- Převod na text: `const char *elf_errmsg(int error)`
  - -1 jako `error` je zkratka pro `elf_errno()`

Typicky:

```
if (! elf_function (...) )
  errx(1, "elf_function: %s", elf_errmsg(-1));
```

Úkol

Inicializace `libelf` a otevření souboru

1. Najděte a otevřete si BSD dokumentaci
2. Upravujte `pb173/04/`
3. Zavolejte `elf_version(EV_CURRENT)`
4. Zkontrolujte návratovou hodnotu
   - Nesmí být `EV_NONE`
   - V případě chyby ji vypište (`elf_errmsg(-1)`)
5. Pomocí standardního `open` otevřete soubor z příkazové řádky
   - Pro čtení
6. Přeložte a spusťte

libelf

Soubory

- `libelf` pracuje s *file deskriptory*
- Začátek práce
  - `Elf *elf_begin(int fd, Elf_Cmd cmd, Elf *ref)`
  - `cmd` je jedno z `ELF_C_READ`, `ELF_C_WRITE`, `ELF_C_RDWR`
  - `ref` je `NULL`
- Ověření typu sobouru
  - `Elf_Kind elf_kind(Elf *elf)`
  - Návratová hodnota je jedno z `ELF_K_NONE`, `ELF_K_AR`, *`ELF_K_ELF`*
- Konec práce
  - `int elf_end(Elf *elf)`

Příklad (bez ověření chyb)

```
int fd = open(file, O_RDONLY);
Elf *elf = elf_begin(fd, ELF_C_READ, NULL);
elf_end(elf);
close(fd);
```

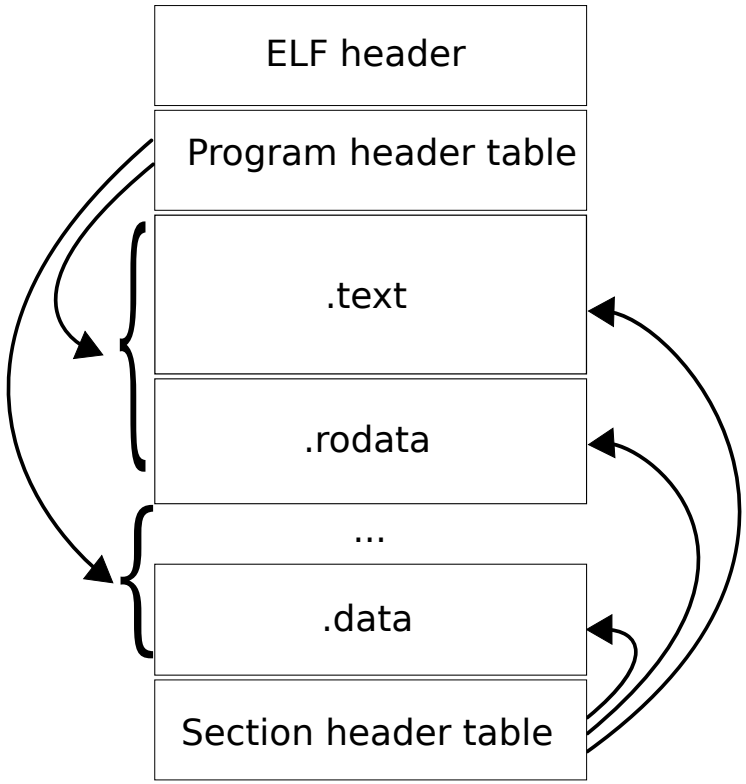
Úkol

Doplňte načtení souboru zadaného jako parametr

1. Zavolejte `elf_begin`
2. Zavolejte `elf_kind`
   - Ověřte, že soubor je typu `ELF_K_ELF`
   - Pokud ne, program ukončete s chybou
3. Zavolejte `elf_end`
4. Ověřujte návratové hodnoty
5. Přeložte a spusťte

Struktura ELFu

- ELF hlavička
- Hlavičky sekcí
  - Popis dat pro linkování, ladění apod.
  - Pro překladač, debugger, . . .
- Hlavičky programové
  - Pohled na data pro spuštění
  - Pro interpretr
- Data
  - Odkazovány z obou typů hlaviček



Zdroj: wikipedia

ELF hlavička

- `readelf -h`
- Magické číslo (`0x7f 'E''L''F'`)
- Typ, cílová architektura, stroj, systém, ...
- Metadata popisující hlavičky
- V `libelf` (`gelf.h`)
  - Struktura `GElf_Ehdr`
  - `gelf_getehdr`

Úkol

Výpis ELF hlavičky

1. Vypište si ELF hlavičku pomocí `readelf`
2. Ze svého programu vypište
   - Velikost ELF hlavičky
   - Architekturu, pro kterou ELF je
   - Počet programových hlaviček
   - Počet hlaviček sekcí
3. Přeložte a spusťte

Sekce

- `readelf -S` (obsah sekce: `readelf -x`, popř. text `readelf -p`)
- 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 při startu na 0
  - `.interp`: interpretr
  - `.symtab`: tabulka symbolů pro ladění (`strip`)
  - `.dynsym`: tabulka symbolů pro dynamický linker
  - `.gnu_debuglink`: odkaz na soubor s ladicími informacemi

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
   - $\Rightarrow$ mějte 2 soubory
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.

- Držátko: `Elf_Scn`
- Hlavička: `GElf_Shdr` (`gelf.h`)
  - Získání hlavičky:
    `GElf_Shdr *gelf_getshdr(Elf_Scn *scn, GElf_Shdr *dst)`
  - `sh_name`: offset do dat sekce `.shstrtab`
  - `sh_type`: `SHT_NULL`, `SHT_PROGBITS`, `SHT_SYMTAB`, `SHT_STRTAB`, ...
  - `sh_flags`: `SHF_WRITE`, `SHF_ALLOC`, `SHF_EXECINSTR`, ...
  - `sh_size`: velikost
- Data: `Elf_Data`
  - Iterace přes obsah:
    `Elf_Data *elf_getdata(Elf_Scn *scn, Elf_Data *data)`
  - `d_buf`: data
  - `d_type`: `ELF_T_BYTE`, `ELF_T_ADDR`, ...
  - `d_size`: velikost dat

libelf

Vyhledání sekcí

- x-tá sekce: `Elf_Scn *elf_getscn(Elf *elf, size_t index)`
- Iterace: `Elf_Scn *elf_nextscn(Elf *elf, Elf_Scn *scn)`
- Index sekce s názvy sekcí `.shstrtab`:
  `int elf_getshdrstrndx(Elf *elf, size_t *dst)`

Příklad

```
Elf_Scn *scn = NULL;
while ((scn = elf_nextscn(elf, scn))) {
  GElf_Shdr shdr;
  Elf_Data *data = NULL;

  gelf_getshdr(scn, &shdr);
  /* shdr */

  while ((data = elf_getdata(scn, data)))
    /* data */;
}
```

Úkol

Hexdump sekcí

1. Iterujte přes sekce (`elf_nextscn`)
2. Vypište hlavičku každé sekce (`gelf_getshdr`)
   - Index (`elf_ndxscn`)
   - Offset do sekce názvů
   - Velikost
   - Flagy
3. Vypište data každé sekce (`elf_getdata`)
   - Uvažte jen první `elf_getdata` (neiterujte přes data)
   - Hexdump prvních 16 bytů obsahu
4. Přeložte a vyzkoušejte
5. Porovnejte s `readelf -S`

- Otevření s `ELF_C_WRITE`/`ELF_C_RDWR`
- Nová sekce: `Elf_Scn *elf_newscn(Elf *elf)`
  - `gelf_getshdr` jako předtím a naplnění
  - Po naplnění:
    ```
    int gelf_update_shdr(Elf_Scn *scn, GElf_Shdr *src)
    ```
  - Vytvoření dat: `Elf_Data *elf_newdata(Elf_Scn *scn)`
- Zápis do souboru
  - `loff_t elf_update(Elf *elf, Elf_Cmd cmd)`
  - `cmd`: `ELF_C_WRITE`

Úkol

Přidání sekce do ELFu/ověření (ne)funkčnosti `libelf`

1. Otevřete ELF soubor pro čtení a zápis
2. Vytvořte novou sekci `.comment.my`
   - Typu `SHT_NOTE`
3. Vložte do ní nějaká data
   - Typu `ELF_T_BYTE`
4. Zavolejte `elf_update`
   - S parametrem `ELF_C_WRITE`
5. Přeložte a vyzkoušejte
6. Zobrazte obsah sekce `.comment.my` pomocí `readelf -x`