PB173 – Binární programování Linux

V. DWARF

Jiri Slaby

ITI, Fakulta informatiky

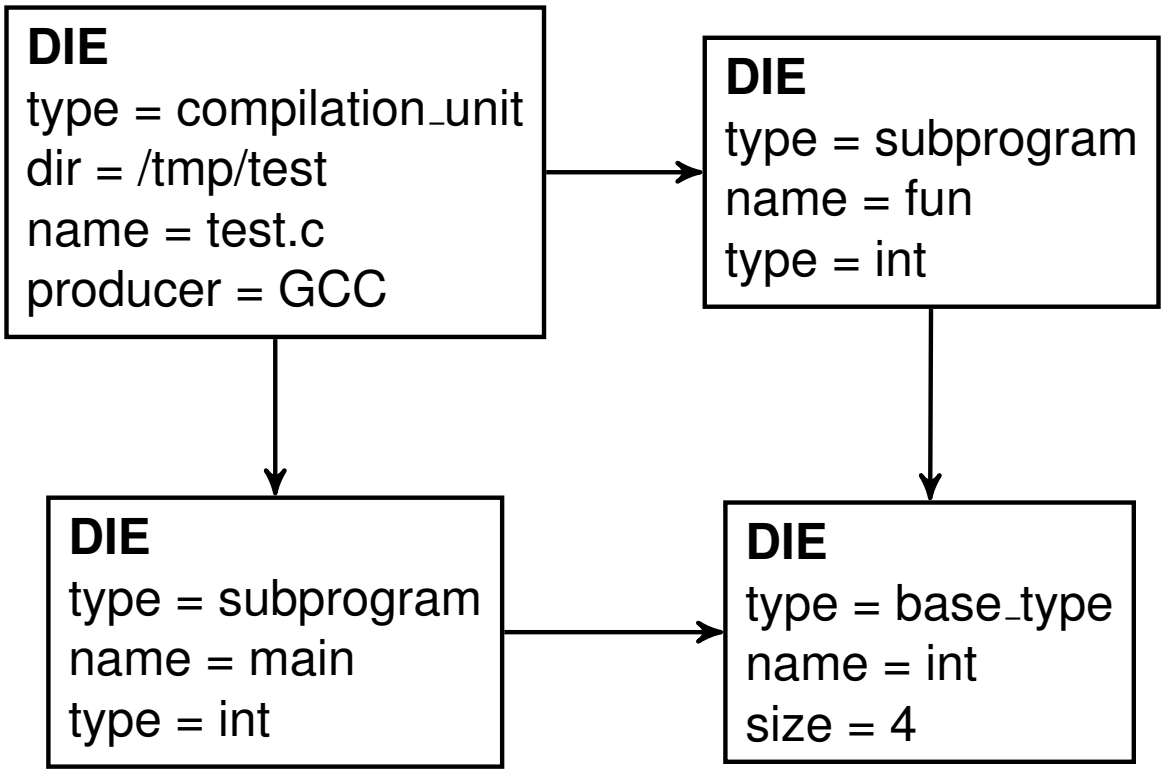
15. 10. 2013

DWARF

- Strukturované ladicí informace
  - Zápis komplikovaných výrazů
  - Vytvářené překladačem
- Nezávislost na jazyku a formátu souborů
- Dokumentace
  - Introduction to the DWARF Debugging Format
  - DWARF Debugging Information Format (Version 4)

DWARF formát

- Ladicí informace v lese skoro-stromů
  - Pro každý zdrojový soubor jeden skoro-strom
- Uzly stromů jsou *Debugging Information Entry* (DIE)
- Jeden DIE může odkazovat na kterýkoliv jiný



DWARF a ELF

1. DWARF je v ELFu uložený v `.debug_*` sekcích
   - `.debug_info`: všechny DIE z lesa stromů
   - `.debug_abbrev`: formát `.debug_info` (Figure 48 v DWARF)
   - `.debug_types`: popis datových typů
   - `.debug_str`: řetězce odkazované z `.debug_info`
   - `.debug_line`: překlad instrukce $\leftrightarrow$ řádek kódu
2. Výpis DWARFu v ELFu
   - `readelf --debug-dump`
   - `objdump --dwarf`
   - `dwarfdump` (z `libdwarf`)

Knihovny na práci s DWARF

Dvě různé implementace knihoven

1. `libdwarf`
   - Starší
   - Hůř se s ní pracuje
   - Obsahuje méně chyb
   - Lépe dokumentovaná
   - `libdwarf/libdwarf.h`
2. `libdw` a `libdwfl`
   - Podpora nových standardů DWARF
   - Lepší rozhraní
   - Základní: `libdw` (`elfutils/libdw.h`)
   - Nadstavba: `libdwfl` (`elfutils/libdwfl.h`)
   - Tuto budeme používat

libdw a libdwfl

- Knihovny jsou součástí elfutils
- Lze je používat střídavě (jako `elf` a `gelf`)
- Inicializace: `Dwfl *dwfl_begin(const Dwfl_Callbacks *cb)`
  - `.section_address = dwfl_offline_section_address`
- Ukončení: `void dwfl_end(Dwfl *dwfl)`
- Poslední chyba: `dwfl_errmsg(-1)`
- Dokumentace
  - V hlavičkových souborech
- Knihovny při překladu: `gcc ... -ldw`

Úkol

Inicializace `libdwfl`

1. Nainstalujte si `libdw` (a `libdwfl`)
2. Vytvořte si `main`
3. Definujte si statické háčky
   - Struktura `const Dwfl_Callbacks`
   - `.section_address = dwfl_offline_section_address`
4. Zavolejte `dwfl_begin`
5. Zavolejte `dwfl_end`
6. Ověřujte návratové hodnoty a vypisujte chyby
7. Přeložte a spusťte

libdw a libdwfl – soubory

Načtení ELFu s DWARFem uvnitř

- `Dwfl_Module *dwfl_report_offline(Dwfl *dwfl, const char *name, const char *file_name, int fd)`
  - `dwfl`: návratová hodnota z `dwfl_begin`
  - `name`: cokoliv (např. "")
  - `file_name`: jméno souboru
  - `fd`: -1 (nebo souborový deskriptor, pak `file_name == ""`)

Průchod stromů (jednotlivých Compilation Unit)

- `Dwarf_Die *dwfl_nextcu(Dwfl *dwfl, Dwarf_Die *lastcu, Dwarf_Addr *bias)`
  - `dwfl`: návratová hodnota z `dwfl_begin`
  - `lastcu`: poslední CU
  - `bias`: přičítá se k adresám, které vrací `libdw`

```
mod = dwfl_report_offline (dwfl, "", argv[1], -1);
while ((die = dwfl_nextcu(dwfl, die, &bias))) { ... }
```

Úkol

Otevření ELFu pomocí `libdwfl`

1. Přidávejte kód mezi `dwfl_begin` a `dwfl_end`
2. Zavolejte `dwfl_report_offline`
3. Iterujte přes CU DIE `dwfl_nextcu`
   - Spočítejte z kolika souborů je objekt přeložen
4. Ověřujte návratové hodnoty a vypisujte chyby
5. Přeložte a spusťte

Debugging Information Entry

- Typ/tag `DW_TAG_*`
- Z DIE: `int dwarf_tag (Dwarf_Die *die)`
  - `DW_TAG_compile_unit`: o souboru
  - `DW_TAG_subprogram`: o funkci
  - `DW_TAG_base_type`: o typu
  - ...
- Seznam dvojic
  - Typ hodnoty: `DW_AT_*`
    - `DW_AT_name`: jméno (proměnné, souboru, ...)
    - `DW_AT_decl_file`: soubor s deklarací
    - `DW_AT_decl_line`: řádek s deklarací
    - `DW_AT_type`: typ proměnné
    - ...
  - Hodnota

DIE
type = compilation_unit
dir = /tmp/test
name = test.c
producer = GCC

Hlavní DIE

```
DW_TAG_compile_unit
    DW_AT_producer          GNU C 4.7.2 20130108 [gcc-4_7-branch revision 195012]
    DW_AT_language          DW_LANG_C89
    DW_AT_name              test.c
    DW_AT_comp_dir          /tmp/test
    DW_AT_low_pc            0x00000000
    DW_AT_entry_pc          0x00000000
    DW_AT_stmt_list         0x00000000
 ...
DW_TAG_subprogram
    DW_AT_external          yes(1)
    DW_AT_name              main
    DW_AT_decl_file         0x00000001 /tmp/test/test.c
    DW_AT_decl_line         0x00000019
    DW_AT_prototyped        yes(1)
    DW_AT_type              <0x0000005b>
    DW_AT_low_pc            0x00000000
    DW_AT_high_pc           0x0000011d
    DW_AT_frame_base         ...
    DW_AT_GNU_all_call_sites yes(1)
    DW_AT_sibling           <0x00000770>
```

Atributy DIE

Iterace přes všechny atributy v jednom DIE

- `ptrdiff_t dwarf_getattrs(Dwarf_Die *die, int (*callback)(Dwarf_Attribute *, void *), void *arg, ptrdiff_t offset)`
  - `die`: z `dwfl_nextcu`
  - `callback`: vaše funkce, která se zavolá pro každý atribut v DIE
  - `arg`: něco, co se předá do `callback` jako 2. parametr
  - `offset`: 0

Operace nad atributem

- Typ: `dwarf_whatattr`
  - Vrací `DW_AT_*` (`DW_AT_name` apod.)
- Typ obsahu: `dwarf_whatform`
  - Vrací `DW_FORM_*` (`DW_FORM_string` apod.)
- Obsah: `dwarf_form*`
  - Např. `dwarf_formstring`

Atributy DIE – příklad

```
static int cb(Dwarf_Attribute *attr, void *ptr)
{
   printf ("AT=%x FORM=%x\n", dwarf_whatattr(attr), dwarf_whatform(attr));
   if (dwarf_whatform(attr) == DW_FORM_strp)
      printf ("name=%s\n", dwarf_formstring(attr));
  return DWARF_CB_OK;
}

...

  while ((die = dwfl_nextcu(dwfl, die, &bias))) {
    dwarf_getattrs(die, cb, NULL, 0);
  }
```

Úkol

Výpis atributů

1. Definujte funkci (háček) pro `dwarf_getattrs`
2. V ní vypište informace
   - Typ atributu (`dwarf_whatattr(attr)`)
   - Typ obsahu atributu (`dwarf_whatform(attr)`)
   - Adresu, pokud se jedná o typ `DW_AT_low_pc`
   - Řetězec a typ atributu `DW_AT_*`, pokud se jedná o typ hodnoty `DW_FORM_string` nebo `DW_FORM_strp`
3. V těle cyklu `dwfl_nextcu` zavolejte `dwarf_getattrs`
4. Přeložte a spusťte

Průchod skoro-stromem

Iterace přes všechny atributy v jednom DIE

- `int dwarf_child(Dwarf_Die *die, Dwarf_Die *result)`
  - Do `result` vloží potomka `die`
  - Vrací 0 (OK), 1 (žádný potomek), -1 (chyba)
  - `result` musí být ukazatel na lokální proměnnou
- `int dwarf_siblingof(Dwarf_Die *die, Dwarf_Die *result)`
  - Do `result` vloží sourozence `die`
  - Vrací 0 (OK), 1 (žádný sourozenec), -1 (chyba)

Úkol

Výpis potomků CU

1. Vezměte 1. potomka od CU (`dwarf_child`)
2. Iterujte přes sourozence potomka (`dwarf_siblingof`)
3. Pro všechny potomky vypište typ DIE (`dwarf_tag`)
   - Nejprve zjistěte, které tam jsou
   - Potom vypisujte v textové podobě
4. Rozšiřte na celý podstrom, nejen první úroveň
5. Přeložte a spusťte