Nízkoúrovňové programování

Linkování

Petr Krajča

Katedra informatiky
Univerzita Palackého v Olomouci

Typy linkování

statické

- linkovaný kód je znám v době překladu
- závislosti jsou součástí spustitelného souboru (výhoda i nevýhoda)
- nezávislost na běžícím prostředí (kritické nasazení, certifikace, apod.)
- aktualizace knihovny vyžaduje opětovný překlad
- nulová režie, možnost optimalizací

dynamické

- linkovaný kód je načten a provázán až za běhu
- sdílení kódu mezi procesy, snazší aktualizace
- režie načítání knihoven a volání funkcí

Ukázka (1/2)

- většinou se implicitně používá dynamické linkování
- ve spustitelném kódu chybí implementace funkcí (`printf`, `sqrt`), viz `objdump -d -M intel sqrt`
- velikost souboru `sqrt` řádově desítky kB
- zjištění závislostí

```
$ ldd sqrt
linux-vdso.so.1 (0x00007ffdf51b0000)
libm.so.6 => /lib64/libm.so.6 (0x00007f1a5209d000)
libc.so.6 => /lib64/libc.so.6 (0x00007f1a51ea2000)
/lib64/ld-linux-x86-64.so.2 (0x00007f1a5219d000)
```

Ukázka (2/2)

- statické linkování lze vynutit přepínačem gcc `-static`
- spustitelný kód obsahuje implementace funkcí (`printf`, `sqrt`), viz `objdump -d -M intel sqrt`
- velikost souboru `sqrt` řádově jednotky MB
- zjištění závislostí

```
$ldd sqrt-static
not a dynamic executable
```

Sestavení dynamicky linkované knihovny

- hlavičkový soubor (obvykle v samostatném adresáři `include`)
- v Linuxu *shared object file*
- název ve tvaru `lib<NAZEV>.so`
- jednotlivé objektové soubory musí být přeloženy jako *position independent code* (PIC), přepínač gcc `-fpic`
- vygenerování souboru s přepínačem `-shared`

```
gcc -shared -o libmyfuncs.so primes.o funcs.o
```

Sestavení staticky linkované knihovny

- hlavičkový soubor (obvykle v samostatném adresáři `include`)
- archiv objektových souborů
- název ve tvaru `lib<NAZEV>.a`
- sestavení archivu programem `ar`

```
ar -rc libmyfuncs_static.a primes.o funcs.o
```

- vytvoření indexu symbolů programem `ranlib`

```
ranlib libmyfuncs_static.a
```

Použití (1/2)

hlavičkový soubor

- `#include "myfuncs.h"`
- pokud není ve výchozí (systémové) cestě, je potřeba překladači předat cestu k hlavičkovým souborům, přepínač `-I`
- `-I../myfuncs/include`

linkování

- přepínač `-l<NAZEV>` (bez předpony `lib` nebo přípony `.a` nebo `.so`)
- pokud není knihovna ve standardní cestě, je potřeba předat překladači cestu ke knihovnám, přepínač `-L`
- `-L../myfuncs/`

Použití (2/2)

spuštění programu

- statická linkování z principu bez komplikací
- u dynamického linkování nutné znát cestu ke knihovně
  - systémové adresáře (např. `/usr/lib64`, `/lib64`)
  - globální proměnná `LD_LIBRARY_PATH`, např.
    `export LD_LIBRARY_PATH=../myfuncs/`

Dynamicky náhravané knihovny

- podobné dynamicky linkovaným knihovnám
- nahrávaná knihovna i volané funkce jsou určeny až za běhu
- vhodné pro implementaci např. pluginů nebo rozšiřujích modulů
- využívají (shared object files)

použití

- funkce `dlopen` – vrátí ukazatel pro práci s knihovnou

```
void *myfuncs_lib = dlopen("libmyfuncs.so", RTLD_LAZY);
```

- funkce `dlsym` – vrátí ukazatel na zvolenou funkci

```
unsigned int (*fib)(unsigned int) = dlsym(myfuncs_lib, "fib");
printf("%i", fib(30));
```

- funkce `dlclose` – ukončení práce s knihovnou

Hodnocený úkol

- sadu funkcí z minulého semináře převeďte na staticky a dynamicky linkovanou knihovnu
- upravte testovácí kód, aby použil oba typy knihoven
- vytvořte program `gonio`, který
  - bude mít dva vstupní argumenty název goniometrické funkce (sin, cos, tan) a úhel ve stupních,
  - vypíše hodnotu dané funkce pro daný úhel,
  - a bude používat dynamické nahrávání knihovny (`libm.so`)