Operační systémy 1

2. Cvičení

Radek Janoštík

KATEDRA
INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

Úkoly z minula – jak to šlo?

- Kolik se vám toho podařilo udělat?
- Dalo něco zabrat?
- Nějaké dotazy?

Úkoly z minula – jak to šlo?

- Kolik se vám toho podařilo udělat?
- Dalo něco zabrat?
- Nějaké dotazy?
- Prezenčka

Překlad programu v jazyce C

- Program v jednom `.c` souboru přeložíme jedním příkazem

```
1       gcc main.c -o main
```

- Ten ale provede více operací:
  - Spuštěn preprocesor (hlavičkové soubory, komentáře, makra)
  - Přeložení kódu v C do jazyka symbolických adres
  - Vytvoření objektového souboru (strojový kód, konstanty, symboly, funkce, . . .)
  - Sloučení s knihovnami a vytvoření binárního souboru

Jednotlivé fáze

- Preprocesor: `gcc -E main.c`
- Překlad do jazyka symbolických adres: `gcc -S main.c`
- Vytvoření objektového souboru `gcc -c main.c`
  - Zobrazení výsledku: `hexdump -C main.o`
  - `objdump -s main.o`
  - Zpětný překlad *disassembling*: `objdump -d -M intel main.o`
- Vytvoření spustitelného souboru: `gcc -o main main.o`

Sestavení většího programu

- Není dobré mít celý program v jednom souboru
  - ⇒ logické členění kódu
  - ⇒ sdílení kódu mezi projekty
  - ⇒ spojování kódů z více jazyků
- Manuálně pak vše překládat je zbytečné a složité ⇒ soubor `Makefile`

```
1 cil: zavislosti
2 <TAB!>prikaz pro sestaveni cile
3 <TAB!>prikaz pro sestaveni cile
```

- Konkrétně

```
1 hello: hello.o
2       gcc -o hello hello.o
3
4 hello.o: hello.c
5       gcc -c hello.c
```

- Překlad jediným příkazem: `make`

Překlad většího programu

- Ukázka kódu a vytvoření Makefile

Programování v assembleru

- Využití assembleru `nasm`

```
1 ; soubor demo.asm
2 global foo
3
4 section .text
5 foo:
6     mov eax, 42
7     ret
```

- Komentáře
- Poskytované symboly
- Sekce
- Návěstí
- Návratová hodnota funkce
- Překlad: `nasm -f elf64 demo.asm`

Spojení Assembleru a C

- Nadeklarujeme prototyp (hlavičku) funkce `foo()`
- Uvedeme soubor `demo.o` jako závislost v Makefile
- Vytvoříme cíl `demo.o` s příkazem pro přeložení

Úkol (1/2)

1. Vyzkoušejte si přeložit ukázkové příklady z textu.
2. Vezměte funkce int2bits a bits2int a umístěte je do samostatného souboru `bits.c` a vytvořte odpovídající hlavičkový soubor `bits.h`.
3. To samé proveďte pro funkce `encode_date` a `decode_date` a ty umístěte do souborů `dates.[ch]`.

Úkol (2/2)

1. Vytvořte program, který výše popsané funkce bude používat.
2. Pro překlad programu vytvořte vhodný *makefile* a program přeložte.
3. S pomocí nástrojů a přepínačů překladače popsaných v přiloženém textu se podívejte na jednotlivé fáze překladu.