Obsah

21 Kompilace, linkování, proměnné

21.1 Překlad zdrojového kódu na stroj

- není možné zdrojový kód přímo spustit → potřeba převést na strojový kód
- kompilace a linkování

Zdrojový kód → cílový kód	Překladový nástroj	Poznámka
jednotka1.c → jednotka1.asm, jednotka2.c → jednotka2.asm, …	kompilátor jazyka C	překlad do assembly či dnes přímo do objektového kódu
jednotka1.asm → jednotka1.obj, jednotka2.asm → jednotka2.obj, …	assembler	vytvoření objektového kódu
jednotka1.obj + jednotka2.obj + … + knihovna1.lib + knihovna2.lib + … → output	linker	spojení objektových kódů a kódu z knihoven do spustitelného kódu

Tab. 21.1: Průběh kompilace

21.1.1 Kompilace

- vytváření objektových souborů z zdrojových souborů
- prováděno překladačem (kompilátor)
  - gcc, clang, intel c++ compiler, mono, Gc, javac OpenJDK…
- několik kroků, např
  - line reconstruction
  - preprocessing
  - lexikální analýza
  - syntaktická analýza / parsování
  - sématická analýza
  - další – optimalizace, analýza, vytváření kódu…
- typy
  - source-to-source – překlad high-level jazyka do high-level jazyka
  - bytecode – kompilace do assembly teoretického stroje
  - just-in-time compilers – kompilace během spuštění; Python, JavaScript, Java, .NET…
  - hardware compilers
  - assembler – převod assembly do strojového kódu
  - …
- spuštění
  - z IDE

  - ruční spuštění příkazů – cmd, bash
  - automatické systémy – make, cmake

21.1.2 Linkování

- seskupení objektových souborů a vytvoření jednoho spustitelného souboru, knihovny nebo objektu
- prováděno linkerem
- hledání dependencies, importování libraries, nastavení entry pointu…
- statické a dynamické linkování
  - statické – všechen potřebný kód v rámci executable, větší velikost souboru
  - dynamické – některé symboly nalezeny až při spuštění programu v operačním systému, menší velikost, potřeba nainstalovat externí knihovny

21.2 Proměnné

- způsob uložení dat v programu
- na hodnotu ukazuje symbolické jméno
- specifikace typu dat – int, string, float, char, bool, double…
- konstanty – keyword `const` – nelze měnit v průběhu programu
- `signed`, `unsigned` – specifikace čísel s/bez znamének, keyword před data typem
- další data typy, než ty zmíněny v tabulce 21.2, definovány ve standardních C knihovnách

Typ	Vysvětlení	Min. velikost (bity)	Formátování
`char`	nejmenší adresovatelná jednotka uchovávající znak; vnitřně číslo	8	`%c`
`signed char`	stejné jako char, garance znaménka, hodnoty $\langle -127, +127 \rangle$	8	`%c`
`unsigned char`	stejné jako char, garance bez znaménka, hodnoty $\langle 0, 255 \rangle$	8	`%c`
`short`, `short int`	krátký integer typ	16	`%hi`, `%hd`
`int`, `signed`, `signed int`	uchování čísla, rozšíření počtu bitů v operačních systémem se širším busem	16	`%i`, `%d`
`unsigned`, `unsigned int`	uchování čísla bez znaménka	16	`%u`
`long`, `long int`	delší `int`, v moderních systémech shodné s `int`	32	`%li` (signed), `%lu` (unsigned)
`long long`, `long long int`	delší `long`	64	`%lli`, `%llu`
`float`	reálné číslo s desetinnými čísly, na většině systémech 32 bitů		`%f`
`double`	větší reálné číslo s desetinnými čísly, na většině systémech 64 bitů		`%lf`
`long double`	velké reálné číslo s desetinnými čísly; 80, 96 nebo 128 bitů		`%Lf`

Tab. 21.2: Data typy v C standartu

21.2.1 Specifikování proměnné v C

- formát „`<datytype> <variable name>;`“
  - `int i;`
  - `char c, ch;`
  - `float f=0.5f, ch;`

21.2.2 Uchování dat v programu

- při specifikování proměnné se alokuje místo v paměti
- čtení/zápis dat následně z paměti na místě adresy
- pointer – adresa proměnné