Úvod do jazyka C

Jan Faigl

Katedra počítačů
Fakulta elektrotechnická
České vysoké učení technické v Praze

Přednáška 7

A0B36PR2 – Programování 2

Úvod do jazyka C

Knihy 1/2

- Učebnice jazyka C, VI. vydání, *Pavel Herout*, KOPP, 2010, ISBN 978-80-7232-406-4

- Učebnice jazyka C – 2. díl, IV. vydání, *Pavel Herout*, KOPP, 2008, ISBN 978-80-7232-367-8

- The C Programming Language, 2nd Edition (ANSI C) , *Brian W. Kernighan, Dennis M. Ritchie*, Prentice Hall, 1988 (1st edition – 1978)

- Algorithms, 4th Edition , *Robert Sedgewick, Kevin Wayne*, Addison-Wesley, 2011, ISBN 978-0321573513

Knihy 2/2

- Programming in C, 4th Edition, *Stephen G. Kochan*, Addison-Wesley, 2014, ISBN 978-0321776419

- C Programming: A Modern Approach, 2nd Edition, *K. N. King*, W. W. Norton & Company, 2008, ISBN 860-1406428577

- 21st Century C: C Tips from the New School, *Ben Klemens*, O'Reilly Media, 2012, ISBN 978-1449327149

- Introduction to Algorithms, 3rd Edition, *Cormen, Leiserson, Rivest, and Stein*, The MIT Press, 2009, ISBN 978-0262033848

Jazyk C

- Nízko-úrovňový programovací jazyk
- Systémový programovací jazyk (operační systém)
- Jazyk pro vestavné (embedded) systémy

  MCU, křížová (cross) kompilace
- Téměř vše nechává na uživateli (programátorovi)

  Inicializace proměnných, uvolňování dynamické paměti
- Má výrazně blíže k využití hardwarových zdrojů

  Přímé volání služeb OS, přímý zápis do registrů a portů.
- Klíčové pro správné fungování programu je zacházení s pamětí

  *Segmentation fault – **valgrind***

Jazyk C nám dává možnost (a někdy povinnost) mít program a jeho překlad plně pod kontrolou.

Zpravidla se vyplatí být obezřetný—základní principy jsou však relativně jednoduché.

Zápis programu

- Zdrojový kód programu v jazyce C se zapisuje do textových souborů
  - **zdrojový** soubor s koncovkou **.c**

    Zpravidla—základní rozlišení souborů, pozor na .C
  - **hlavičkový** soubor s koncovkou **.h**

    Jména souborů volíme výstižné (krátké názvy) a zpravidla zapisujeme malými písmeny.
- Zdrojové soubory jsou překládány do binární podoby překladačem a vznikají objektové soubory (**.o**)

  Objektový kód obsahuje relativní adresy proměnných a volání funkcí nebo pouze odkazy na jména funkcí, jejichž implementace ještě nemusejí být známy.
- Z objektových souborů (object files) se sestavuje výsledný program, ve kterém jsou již všechny funkce známy a relativní adresy se nahradí absolutními.

  Program se zpravidla sestavuje z více objektových souborů umístěných například v knihovnách.

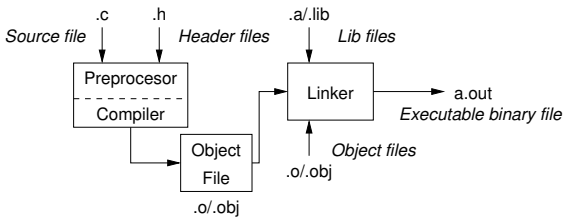
Zdrojové soubory

- Rozdělení na zdrojové a hlavičkové soubory umožňuje rozlišit deklaraci a definici, především však podporuje
  - **Organizaci** zdrojový kódů v adresářové struktuře souborů
  - **Modularitu**
    - Hlavičkový soubor obsahuje popis co modul nabízí
    - Popis (seznam) funkcí a jejich parametrů bez konkrétní implementace
  - **Znovupoužitelnost**
    - Pro využití binární knihovny potřebuje znát její „rozhraní", které je definované v hlavičkovém souboru

Překlad a sestavení programu

- Před vlastním překladem se zdrojové soubory předzpracují **preprocesorem**

  Všechny odkazované hlavičkové soubory se vloží do jediného zdrojového souboru
- Zdrojový soubor se přeloží do objektového souboru
- Výsledný spustitelný soubor se sestaví z příslušných dílčích objektových souborů a odkazovaných knihoven



Části překladu a sestavení programu

- **preprocesor** – umožňuje definovat makra a tím přizpůsobit překlad aplikace kompilačnímu prostředí

  Výstupem je textový („zdrojový") soubor.
- **compiler** – Překládá zdrojový (textový) soubor do strojově čitelné (a spustitelné) podoby

  Nativní (strojový) kód platformy, bytecode, případně assembler
- **linker** – sestavuje program z objektových souborů do podoby výsledné aplikace

  Stále může odkazovat na knihovní funkce (dynamické knihovny linkované při spuštění programu), může též obsahovat volání OS (knihovny).
- Dílčí části **preprocesor**, **compiler**, **linker** jsou zpravidla „jediný" program, který se volá s příslušnými parametry

Překladače jazyka C

- V rámci předmětu PR2 budeme používat především překladače z rodin:
  - `gcc` – GNU Compiler Collection
    `https://gcc.gnu.org`
  - `clang` – C language family frontend for LLVM
    `http://clang.llvm.org`

Pro win platformy pak odvozená prostředí cygwin `https://www.cygwin.com/` nebo MinGW `http://www.mingw.org/`*

- Základní použití (přepínače a argumenty) je u obou překladačů stejné

clang je kompatibilní s gcc

- Příklad použití
  - compile: `gcc -c main.c -o main.o`
  - link: `gcc main.o -o main`

Příklad triviálního C programu

- Spustitelný program musí obsahovat právě jednu definici funkce `main`
- Při spuštění programu předává operační systém programu počet argumentů (`argc`) a argumenty (`argv`)

Pokud používáme OS

  - Prvním argumentem je jméno spouštěného programu

```
1 int main(int argc, char **argv) {
2    int v;
3    v = 10;
4    v = v + 1;
5    return argc;
6 }
```

`lec07/var.c`

Překlad a příklad spuštění programu

- Překlad programu překladačem `clang` – automaticky dojde ke kompilaci a linkování programu do soubor `a.out`
  clang var.c
- Překlad programu do soubor `var`
  clang var.c -o var
- Spuštění programu
  ./var
- Překlad a spuštění
  clang var.c -o var; ./var
- Překlad a spuštění pouze pokud překlad proběhl v pořádku
  clang var.c -o var && ./var

Programy vrací návratovou hodnotu—0 znamená v pořádku

Logický operátor dle použitého interpretu příkazů (např. sh, bash, zsh).

Příklad spuštění programu

- Navratová hodnota programu je uložena v proměnné **$?**

sh, bash, zsh

- Příklad spuštění programu s různým počtem argumentů

```
./var

./var; echo $?
1

./var 1 2 3; echo $?
4

./var a; echo $?
2
```


Příklad zdrojového souboru po zpracování preprocesorem

- Přepínačem **-E** můžeme provést pouze zpracování zdrojového souboru preprocesorem

`gcc -E var.c`

Alternativně též `clang -E var.c`

```
1  # 1 "var.c"
2  # 1 "<built-in>"
3  # 1 "<command-line>"
4  # 1 "var.c"
5  int main(int argc, char **argv) {
6     int v;
7     v = 10;
8     v = v + 1;
9     return argc;
10 }
```


Příklad zdrojového souboru přeloženého do Assembleru

- Přepínačem **-S** můžeme zdrojový kód přeložit do Assembleru

`clang -S var.c -o var.s`

```
1    .file "var.c"
2    .text
3    .globl  main
4    .align  16, 0x90
5    .type main,@function
6  main:                   # @main
7    .cfi_startproc
8  # BB#0:
9    pushq %rbp
10 .Ltmp2:
11   .cfi_def_cfa_offset 16
12 .Ltmp3:
13   .cfi_offset %rbp, -16
14   movq  %rsp, %rbp
15 .Ltmp4:
16   .cfi_def_cfa_register %rbp
17   movl  $0, -4(%rbp)
18   movl  %edi, -8(%rbp)
19   movq  %rsi, -16(%rbp)
20   movl  $10, -20(%rbp)
21   movl  -20(%rbp), %edi
22   addl  $1, %edi
23   movl  %edi, -20(%rbp)
24   movl  -8(%rbp), %eax
25   popq  %rbp
26   ret
27 .Ltmp5:
28   .size main, .Ltmp5-main
29   .cfi_endproc
30
31
32   .ident  "FreeBSD clang
       version 3.4.1 (tags/
       RELEASE_34/dot1-final
       208032) 20140512"
33   .section  ".note.GNU-stack","
       ",@progbits
```


Příklad překladu souboru

- Překlad do objektového souboru

`clang -c var.c -o var.o`

```
% clang -c var.c -o var.o
% file var.o
var.o: ELF 64-bit LSB relocatable, x86-64, version 1
    (FreeBSD), not stripped
```

- Linkování objektového souboru do spustitelného souboru

`clang var.o -o var`

```
% clang var.o -o var
% file var
var: ELF 64-bit LSB executable, x86-64, version 1 (
    FreeBSD), dynamically linked (uses shared libs),
    for FreeBSD 10.1 (1001504), not stripped
```

dynamically linked
not stripped

Příklad spustitelného souboru 1/2

- Standardně je při překladu binární soubor stále vázán na knihovny jazyka C (a služby operačního systému)
- Závislosti můžeme zobrazit voláním `ldd var`

```
ldd var                                  ldd – list dynamic object dependencies
var:
        libc.so.7 => /lib/libc.so.7 (0x2c41d000)
```

- Statické linkování můžeme vynutit přepínačem `static`

```
clang -static var.o -o var
% ldd var
% file var
var: ELF 64-bit LSB executable, x86-64, version 1 (
    FreeBSD), statically linked, for FreeBSD 10.1
    (1001504), not stripped
% ldd var
ldd: var: not a dynamic ELF executable
```

Porovnejte výslednou velikost souborů!

Příklad spustitelného souboru 2/2

- Přeložený program (objektový soubor) standardně obsahuje symbolická jména

Vhodná například pro ladění programu, ladění viz další přednášky.

```
clang var.c -o var
wc -c var
    7240 var
```

wc – word, line, character, and byte count
-c – byte count

- Symboly můžeme odstranit nástrojem (programem) **strip**

```
strip var
wc -c var
    4888 var
```

Alternativně lze velikost souboru zobrazit například příkazem `ls -l`

Knihovní funkce

- Jazyk C sám o sobě poskytuje relativně jednoduchou syntax a sémantiku
- V zásadě umožňuje definovat proměnné, funkce, cykly a výrazy
- V podstatě pro každou trochu složitější činnost je nutné „importovat" (*includovat*) knihovní funkce
- Základní knihovny (std) jsou součástí vývojového prostředí (překladače)

  Viz výpis závislosti na knihovně libc.so.7
- Knihovní funkce se importují klíčovým slovem **preprocesoru** `#include` a uvedením jména hlavičkového souboru knihovny uzavřené ve dvojici znaků
  - `<` a `>` pro systémové knihovny
  - `"` a `"` pro vlastní hlavičkové soubory modulů
- Při překladu specifikujeme příslušné prohledávané adresáře přepínačem `-I` a uvedením cesty

  Pro standardní systémové adresáře není třeba.

Na pořadí uvedení cest a dále pak souborů při linkování záleží!

Příklad programu s výstupem na `stdout`

- Pro výpis na standardní výstup použijeme příkaz formátovaného výstupu `fprintf` z knihovny `stdio.h`

```
1  #include <stdio.h>
2
3  int main(int argc, char **argv) {
4   fprintf(stdout, "My first program in C!\n");
5   fprintf(stdout, "Its name is \"%s\"\n", argv[0]);
6   fprintf(stdout, "Run with %d arguments\n", argc);
7   if (argc > 1) {
8    fprintf(stdout, "The arguments are:\n");
9    for(int i = 1; i < argc; ++i) {
10    fprintf(stdout, "Arg: %d is \"%s\"\n", i, argv[i]);
11   }
12  }
13 }
```


Příklad spuštění

- V případě zahrnutí hlavičkového soubor `stdio.h` jsou „includovány" další soubory, dále jsou definovány příslušné typy a deklarovány další funkce pro vstup a výstup

Ověřte např. `clang -E print_args.c`

```
clang print_args.c -o print_args
./print_args first second
My first program in C!
Its name is "./print_args"
It has been run with 3 arguments
The arguments are:
Arg: 1 is "first"
Arg: 2 is "second"
```


Shrnutí přednášky

Diskutovaná témata

- Jazyk C – překlad zdrojových kódů a linkování programu
- Příklad překladu programu
- Příklad programu s výstupem na standardní výstup

- Příště: základní typy, řídicí struktury, řetězce, pole a ukazatele.