

PB071 – Programování v jazyce C

Úvod, organizace, nástroje

1

Cíle předmětu

1. Zavést a podpořit programátorské schopnosti
2. Seznámit s možnostmi jazyka C
3. Používat základní vývojové nástroje
4. Trochu nadchnout (nebo alespoň úplně neodradit) do programování 😊

Organizační

Historie, normy

Oblasti použití

Začínáme s C

Nástroje

Lehký průlet C

Organizační (1)

● Přednášky

- nepovinné, ale snad přínosné a zábavné ☺
- jedna na **vnitrosemestrální test**
- předpoklad základní znalosti algoritmizace (co je cyklus...)
- rozcestník <http://cecko.eu/public/pb071>

● Cvičení

- povinné, dvouhodinové, dvě neúčasti tolerovány (sem. skupiny)
- aktivní práce na příkladech a domácích úkolech, konzultace
- průběžné testíky (přímo na hodině, za 3 body max.)
- podklady http://cecko.eu/public/pb071_cviceni

● Ukončení předmětu

- **zápočet** – úkoly + průběžný test + testíky, zisk alespoň 65 bodů + úspěšné vypracování zápočtového příkladu na hodině
- **zkouška** – zápočet + zkouškový test, zisk alespoň 95 bodů

Organizační (2)

- Slidy z přednášky, ukázkové zdrojáky
 - http://cecko.eu/public/pb071_cviceni
- Domácí úkoly
 - 5+1 za semestr, zadávány průběžně (na webu cvičení)
 - body za funkčnost, body za správné odevzdání
 - deadline pro odevzdání (na stránce úkolu, 2 týdny)
 - budou zveřejňována ukázková řešení
- Odevzdání/testování
 - možnost odevzdání nanečisto (detaily na cvičení)
 - odevzdání do fakultního SVN, spuštění notifikačního skriptu
 - 12 bodů max. + bonusy (poměr 9 funkčnost, 3 odevzdání)
 - max. 3 pokusy na odevzdání
 - strhávání fixních bodů při nalezení chyby

Neopisujte



- Škodíte především sami sobě
 - začněte pracovat včas, ať máte čas na řešení "záseků"
- Provádíme automatickou kontrolu opisu u všech odevzdaných příkladů
 - každý s každým
 - každý s řešeními z minulých let (pokud je podobný příklad)
 - u podezřelých příkladů probíhá manuální kontrola
- V případě opsání jsou potrestáni oba účastníci

Předpoklady, návaznost na další předměty

- Předpoklady

- předchozí zkušenost s libovolným programovacím jazykem (vlastní nebo IB001)
- základy algoritmizace
- (příkazy, podmínky, cykly, funkce, koncept proměnné)

- Na předmět PB071 navazuje

- PB161 Programování v jazyce C++ (3. semestr)
- PB162 Programování v jazyce Java (3. semestr)
- PB173 Tématické programování C/C++ (3,5 sem.)
- práce v laboratořích (nebojte se zeptat)

- Seznam předmětů s programováním na FI

- <http://www.cecko.eu/public/code@fimu>

7

Kontakt

- Přednášející

- Petr Švenda, svenda@fi.muni.cz
- Konzultační hodiny: Pondělí 14-14:50 G201
 - Gotex, Laboratoř bezpečnosti a aplikované kryptografie

- Cvičící

- na hodinách – využívejte hojně

- Studentští konzultanti

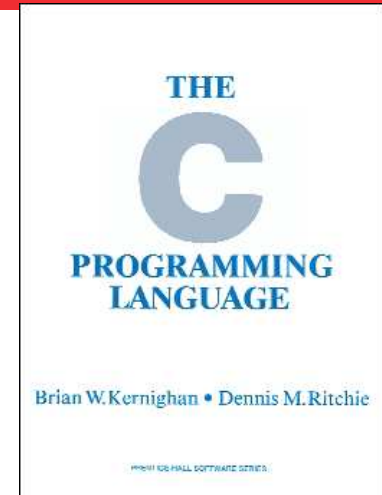
- dodatečné konzultace nezávisle na skupině
- v definované konzultační hodiny (viz. hlavní web)
 - od druhého týdne, bude upřesněno

Sběr zpětné vazby

- Občasné dotazníčky (obtížnost úloh, porozumění...)
- Samozřejmě možné osobně
- Předmětová anketa
 - vyhodnocení PB071 jaro 2013:
http://cecko.eu/public/pb071_hodnoceni_jaro2013
- Velké poděkování všem předem za poznámky a náměty!

Organizační
Historie, normy
Oblasti použití
Začínáme s C
Nástroje
Lehký průlet C

Jazyk C v kontextu



- 1969-73 K-R C (AT&T Bell Labs)

- Brian Kernighan, Dennis Ritchie
- pro systémové programování v rámci UNIXu
- Kniha The C Programming Language (1978)
 - Programovací jazyk C, CPress, 2006

- Imperativní, procedurální, staticky typovaný jazyk

série příkazů měnící
stav programu

podpora strukturovaného
programování

typ (většiny) proměnných
znám v době překladu

- **Není objektově orientovaný**

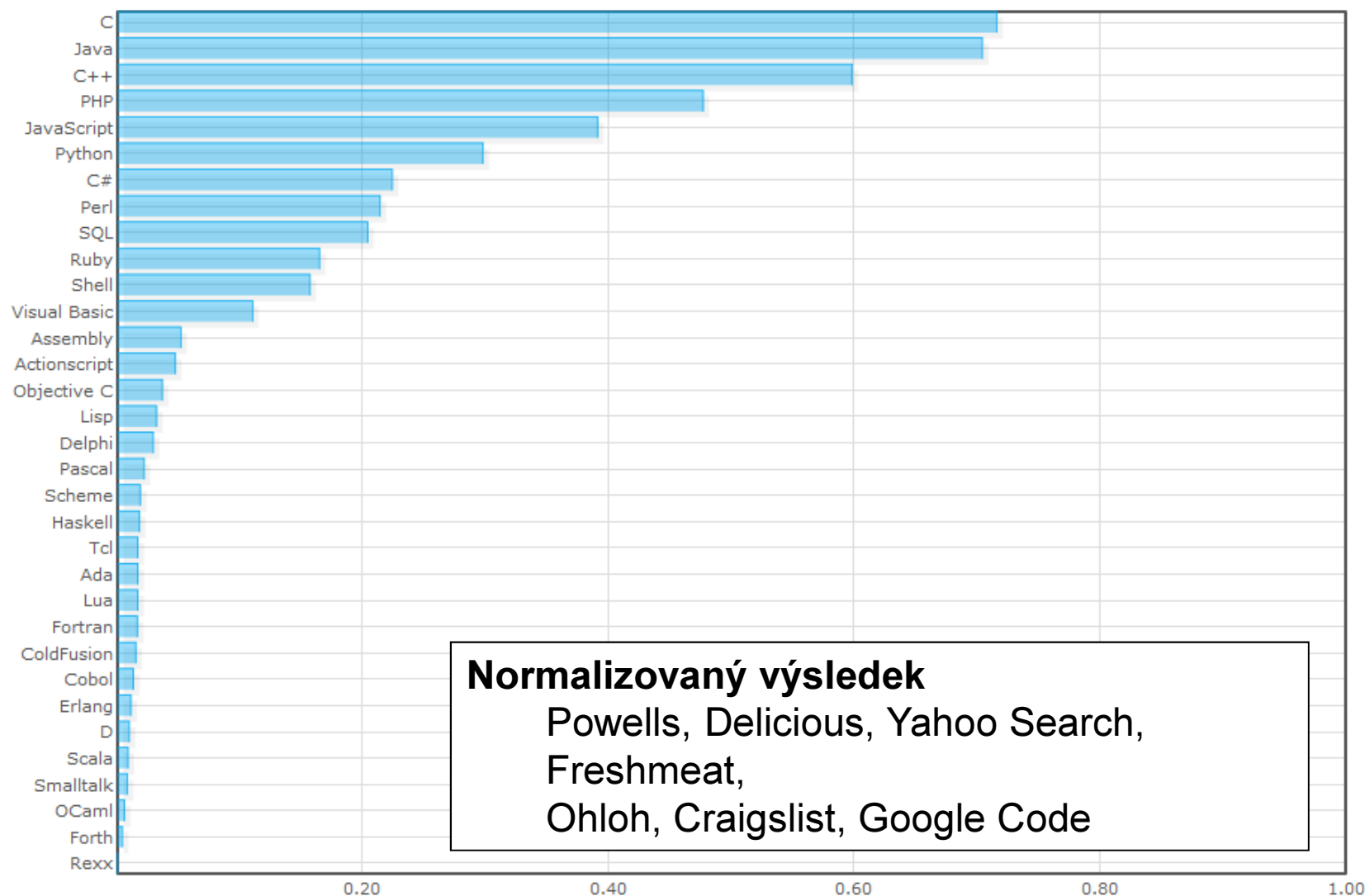
- nejsou přímo jazykové konstrukce (např. typ class)

11

Proč se učit a používat jazyk C

- Nízkoúrovňový jazyk, snadné mapování na strojový kód
 - použití na aplikace původně implementované v assembleru
 - zdrojový kód kompilovaný do nativního kódu HW platformy
 - rychlost, nutnost minimální podpory ze strany běhového prostředí
 - velká kontrola nad prostředím
 - (With great power comes great responsibility ☺)
- Jeden z nejpopulárnějších jazyků vůbec
 - překladač C existuje pro téměř všechny počítačové platformy
 - základ pro syntaxi spousty dalších jazyků
- Typicky vysoká rychlost kódu
 - <http://shootout.alioth.debian.org>

Popularita jazyku - <http://langpop.com/>



13

Srovnání rychlostí – práce s poli

reverse-complement benchmark ~240MB N=25,000,000

This table shows 5 measurements - CPU Time, Elapsed Time, Memory, Code and ~ CPU Load.

		sort	sort	sort	sort	
×	Program Source Code	CPU secs	Elapsed secs	Memory KB	Code B	~ CPU Load
1.0	C++ GNU g++ #4	1.12	1.12	245,432	2275	1% 0% 1% 100%
1.1	ATS	1.18	1.19	122,628	2077	1% 0% 1% 99%
1.2	C++ GNU g++ #2	1.35	1.35	245,080	1098	0% 0% 1% 100%
1.2	C GNU gcc #4	1.38	1.38	125,188	722	0% 0% 2% 100%
1.5	Ada 2005 GNAT #2	1.68	1.70	197,584	3132	1% 0% 1% 99%
1.6	C++ GNU g++ #3	1.75	1.75	125,272	810	0% 0% 1% 100%
2.1	Scala #4	2.37	2.39	400,184	505	0% 0% 0% 99%
2.1	Pascal Free Pascal #2	2.39	2.39	123,816	751	0% 0% 0% 100%
2.6	Java 6 -server #4	2.86	2.90	473,280	592	0% 1% 0% 99%
2.7	C# Mono	3.06	3.05	161,548	1099	0% 0% 0% 100%
3.6	Haskell GHC #2	4.02	4.02	618,032	913	0% 0% 0% 100%
3.8	C++ GNU g++	4.30	4.30	245,388	571	3% 6% 1% 100%
3.9	Lisp SBCL	4.40	4.41	222,396	896	0% 0% 0% 100%
4.3	OCaml #2	4.78	4.78	168,920	394	0% 0% 0% 100%
5.2	Perl #4	5.80	5.80	124,036	237	0% 0% 1% 100%
6.3	PHP #2	7.00	7.00	444,456	343	0% 0% 0% 100%

14

Srovnání rychlostí – matematické operace

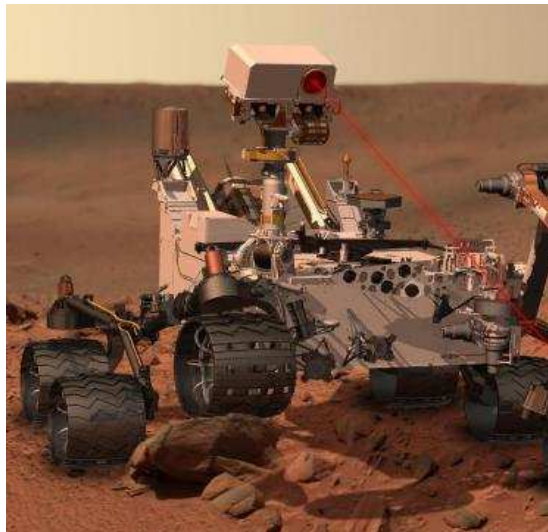
spectral-norm benchmark N=5,500

This table shows 5 *measurements* - CPU Time, Elapsed Time, Memory, Code and ~ CPU Load.

		sort	sort	sort	sort	
×	Program Source Code	CPU secs	Elapsed secs	Memory KB	Code B	~ CPU Load
1.0	C GNU gcc #4	11.87	2.99	772	1139	99% 100% 99% 99%
1.0	C++ GNU g++ #7	11.89	2.99	1,196	1114	100% 99% 99% 99%
1.3	Ada 2005 GNAT #3	15.69	3.98	2,528	1702	98% 99% 98% 99%
1.3	Fortran Intel	15.93	4.00	1,276	568	99% 99% 100% 100%
1.4	Haskell GHC	16.02	4.11	2,260	869	96% 99% 96% 99%
1.4	Java 6 steady state #2	17.14	4.31	24,620	1027	99% 99% 99% 99%
1.5	Java 6 -server #2	17.33	4.51	15,208	950	98% 95% 94% 97%
1.5	Scala #2	17.66	4.56	20,720	720	96% 96% 97% 98%
1.6	Ada 2005 GNAT #2	18.75	4.74	3,012	1464	99% 99% 99% 98%
1.9	C# Mono #2	22.31	5.63	5,104	1063	99% 99% 99% 99%
2.0	ATS #2	22.06	5.96	1,656	2339	92% 92% 93% 93%
2.0	Lisp SBCL #3	22.22	6.01	7,476	883	92% 92% 93% 93%
2.1	OCaml #3	20.02	6.21	3,304	907	79% 79% 81% 81%
2.2	Go 6g 8g #2	26.40	6.63	6,672	545	99% 100% 100% 100%
4.1	Erlang HiPE #2	47.70	12.18	13,348	747	98% 98% 97% 98%

Vhodnost použití jazyka C

- Vhodné využití pro:
 - rychlé vědecké výpočty
 - systémové aplikace
 - programování hardwarových a embedded zařízení
 - rychlá grafika (hry), **rychlost obecně**
- Spíše nevhodné pro
 - webové aplikace (PHP, JavaScript...)
 - rychlé prototypy (ale nutno znát dobře jiný jazyk)
 - větší projekty vyžadující objektově orientovaný návrh (C++, Java...)



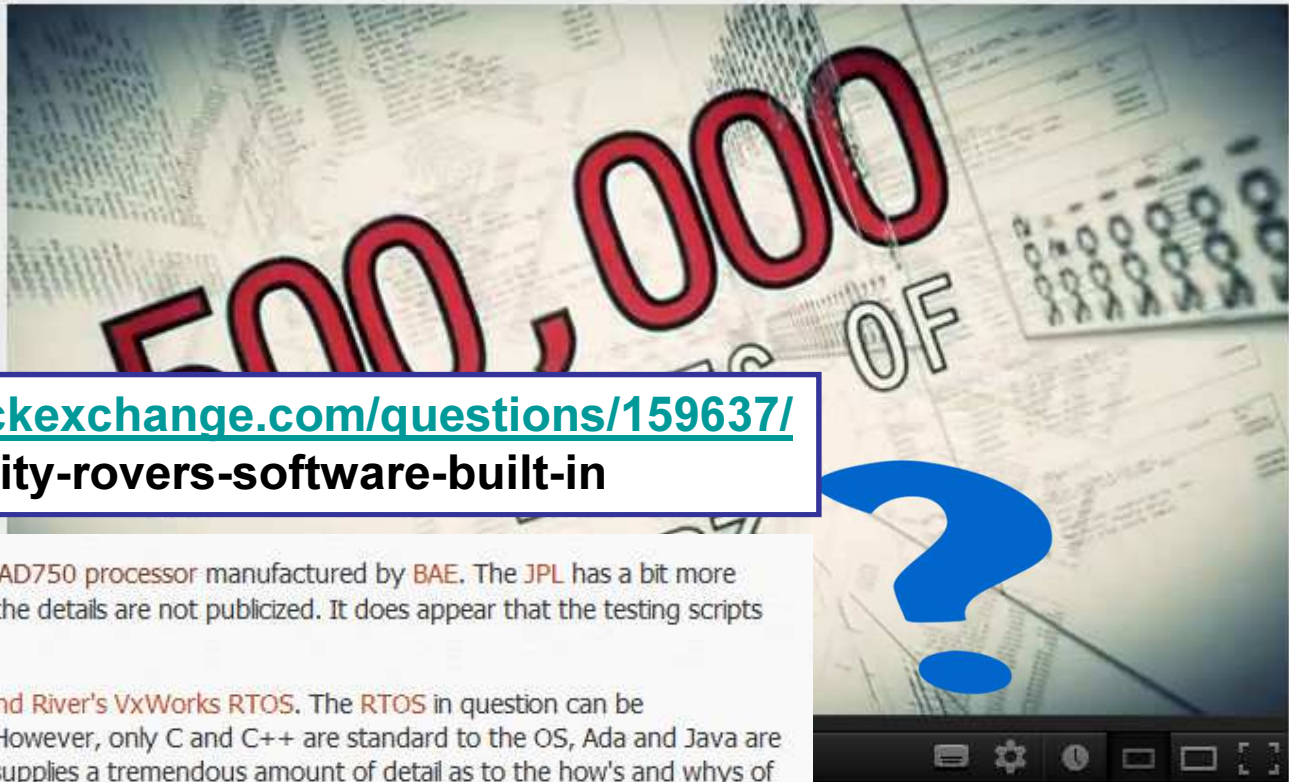
Challenges of Getting to Mars: Curiosity's Seven Minutes of Terror

JPLnews



Subscribe

336 videos



<http://programmers.stackexchange.com/questions/159637/what-is-the-mars-curiosity-rovers-software-built-in>

275

It's running 2.5 million lines of C on a RAD750 processor manufactured by BAE. The JPL has a bit more information but I do suspect many of the details are not publicized. It does appear that the testing scripts were written in Python.



The underlying operating system is Wind River's VxWorks RTOS. The RTOS in question can be programmed in C, C++, Ada or Java. However, only C and C++ are standard to the OS, Ada and Java are supported by extensions. Wind River supplies a tremendous amount of detail as to the how's and whys of VxWorks.

An Erlang programmer talks about the features of the computers and codebase on Curiosity.

share improve this answer

edited Aug 12 at 19:39

answered Aug 6 at 4:30



World Engineer

9,373 3 26 49

29 JPL C language coding standards, specifically for embedded environments instead of "ground software" as they call it. lars-lab.jpl.nasa.gov/JPL_Coding_Standard_C.pdf – Patrick Hughes Aug 6 at 6:21

1,922,565

16,215 likes, 167 dislikes

17

Normy, standardy a rozšíření

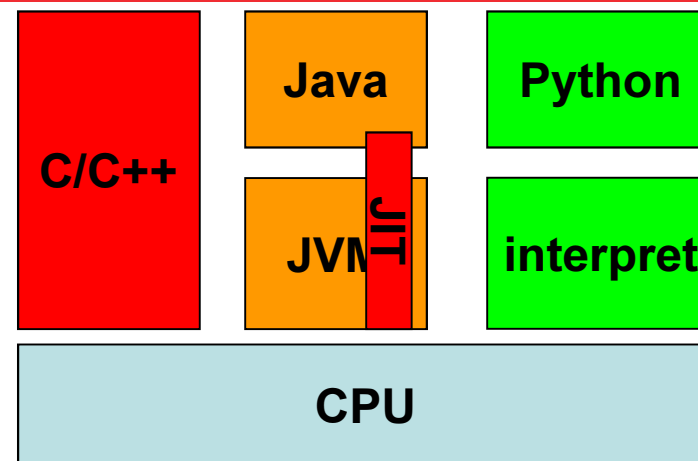
- Kniha The C Programming Language (1978)
 - neformální norma
- ANSI X3.159-1989 (ANSI C, Standard C, C89)
- ISO/IEC 9899:1990 (jen převzaté ANSI C, C90)
- ISO/IEC 9899:1999 (C99)
 - gcc -std=c99
 - (budeme využívat při psaní)
- ISO/IEC 9899:2011 (C11, nejnovější)
 - probereme rozšíření (vlákna, synchronizace...)
 - [http://en.wikipedia.org/wiki/C11_\(C_standard_revision\)](http://en.wikipedia.org/wiki/C11_(C_standard_revision))¹⁸

Nestandardizovaná rozšíření

- Nestandardizované rozšíření
 - užitečné prvky jazyka dosud neobsažené v normě (např. gnu99)
 - specificky označeny a dokumentovány
- Problém: využívání vede k omezení přenositelnosti
 - pro jinou platformu nelze překompilovat bez změny kódu
 - omezuje dostupnost programu
 - zvyšuje cenu přechodu na jinou platformu (customer lock-in)
- Proč psát program v souladu s normou?
 - lze přímo kompilovat pro jiné platformy - svoboda volby platformy
 - svoboda volby kompilátoru a odolnost vůči jeho změnám
 - větší potenciální využití kódu (i jiné projekty/překladače)
 - norma může omezit problematické jazykové konstrukce (nižší chybovost)

Jazyk C a další

- Jazyk C/C++
 - překlad přímo do strojového kódu
 - překlad nutný zvlášť pro každou platformu
- Další imperativní: Java, C#...
 - překlad do mezi jazyku bytecode/CIL
 - jedna binárka pro všechny platformy
 - (Java Virtual Machine) JVM pro velké množství platforem
 - bytecode interpretovaný, ale JIT (Just-In-Time) kompilátor
- Skriptovací imperativní: Perl, Python...
 - typicky se interpretuje, platformově nezávislé (pokud je interpret)
- Funcionální: Haskell, LISP...
 - jiné paradigma: matematický zápis odvození z počátečních hodnot
- Logické programování: Prolog...
 - jiné paradigma: JAK má výsledek vypadat, ne jak se k němu dostat

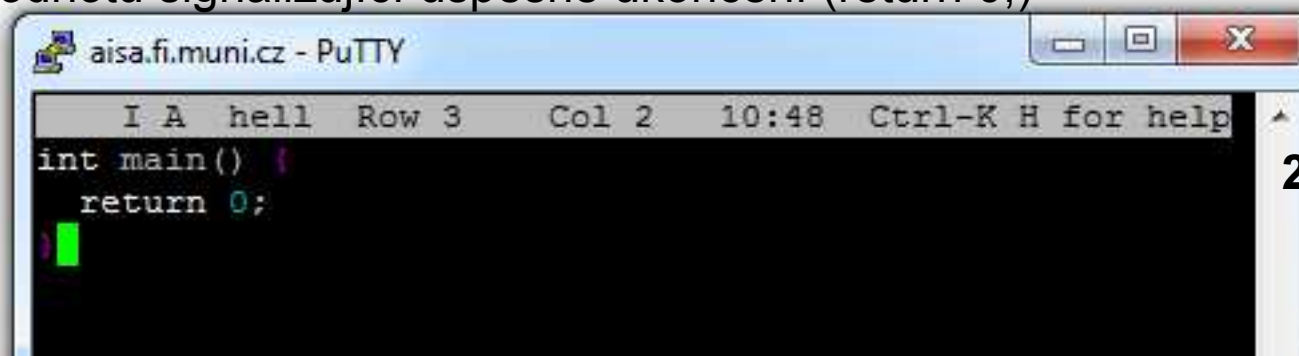


Historie, normy
Oblasti použití
Začínáme s C
Nástroje
Lehký průlet C

Hello World (na Aise) – Pokus 1

1. Připojíme se na Aisu (2x, pro edit & pro překlad)
 - Unix/Linux: `ssh váš_login@aisa.fi.muni.cz`
 - Windows: Putty `váš_login@aisa.fi.muni.cz`
2. Vytvoříme soubor s příponou .c (hello.c)
 - např. `pico` hello.c
3. Vložíme funkci se speciálním jménem `main`
 - návratová hodnota `int` (celé znaménkové číslo - integer)
 - zatím bez parametrů (kulaté závorky)
4. Implementujeme tělo funkce `main`
 - do složených závorek `{}`
 - vrátíme hodnotu signalizující úspěšně ukončení (`return 0;`)
 - uložíme

```
int main() {  
    return 0;  
}
```



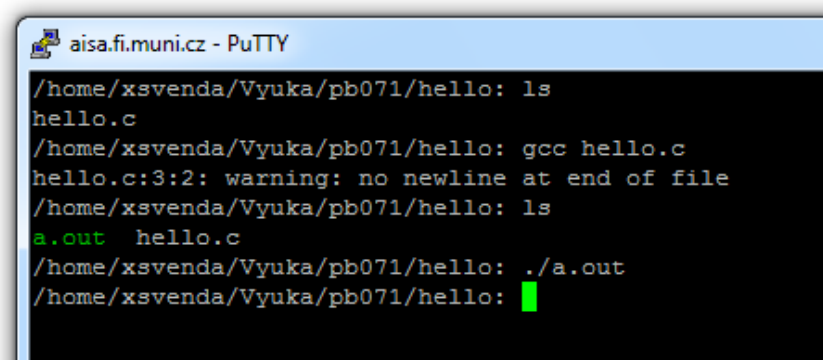
```
aisa.fi.muni.cz - PuTTY  
I A hell Row 3 Col 2 10:48 Ctrl-K H for help  
int main() {  
    return 0;  
}
```

Hello World (na Aise) – Pokus 1 (pokr.)

5. Přeložíme

- `gcc hello.c`
- vznikne soubor `a.out`

6. Spustíme: `./a.out`



```
aisa.fi.muni.cz - PuTTY
/home/xsvenda/Vyuka/pb071/hello: ls
hello.c
/home/xsvenda/Vyuka/pb071/hello: gcc hello.c
hello.c:3:2: warning: no newline at end of file
/home/xsvenda/Vyuka/pb071/hello: ls
a.out hello.c
/home/xsvenda/Vyuka/pb071/hello: ./a.out
/home/xsvenda/Vyuka/pb071/hello: 
```

● Věci ke zlepšení

- nic nevypisuje
- chybí komentáře
- odstranit warning (no newline at end of file)
- překlad starou verzí gcc (gcc --version)
- kontrola shody vůči standardu

23

Hello World (na Aise) – Pokus 2

- Přidání výpisu na standardní výstup
 - typicky konzole, obrazovka
 - funkce `printf` (google: C printf)

- Komentáře

```
#include <stdio.h>

/*
  This is (possibly) multi
  line commentary
*/
int main() {
  // This is single line commentary
  printf("Hello world\n");
  return 0;
}
```

knihovna obsahující funkci
`printf`

klíčové slovo pro vložení
knihovných funkcí

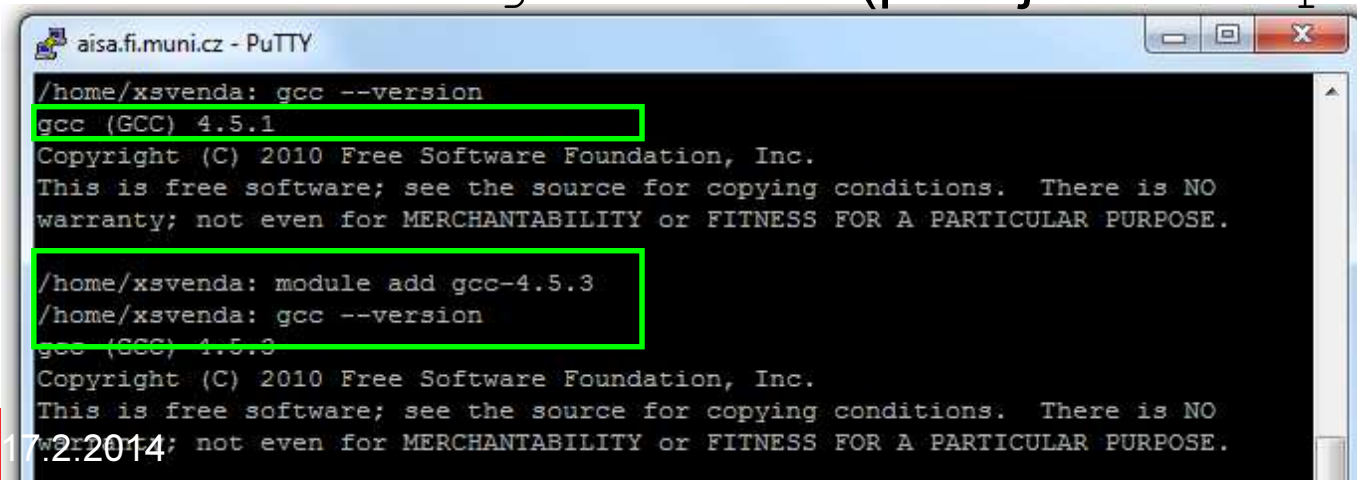
funkce pro vytištění řetězce

parametr funkce `printf`,
řetězec "Hello world"

24

Hello World (na Aise) – Pokus 2 (pokr.)

- Při překladu varování (warning)
 - warning: no newline at end of file
 - přidáme nový řádek na konec zdrojového souboru
- Překlad starou verzí gcc
 - verzi zjistíme pomocí `gcc --version`
 - na Aise defaultně starší verze, budeme používat 4.5.3
 - `module add gcc-4.5.3` (přidejte si do `.profile`)



```
aisa.fi.muni.cz - PuTTY
/home/xsvenda: gcc --version
gcc (GCC) 4.5.1
Copyright (C) 2010 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

/home/xsvenda: module add gcc-4.5.3
/home/xsvenda: gcc --version
gcc (GCC) 4.5.3
Copyright (C) 2010 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

25

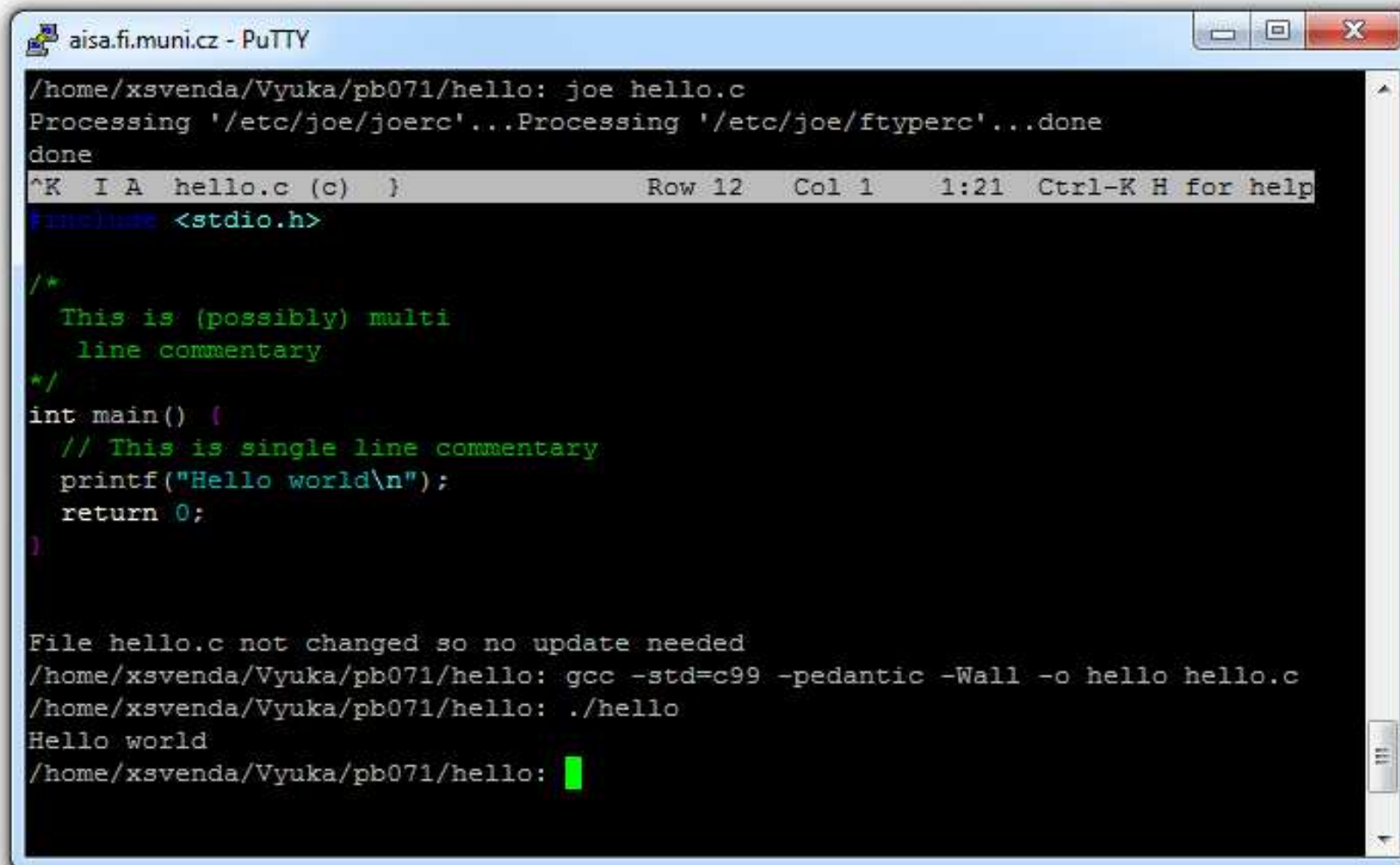
Hello World (na Aise) – Pokus 2 (pokr.)

- Kontrola shody vůči standardu
 - různé překladače mají různý stupeň podpory
 - několik verzí standardu, lze kontrolovat vůči konkrétní
- Přepínače překladače
 - `gcc hello.c (->a.out)`
 - *default* `-std=gnu99` (C99 + GNU rozšíření)
 - `gcc -std=c99 -pedantic -Wall -o hello hello.c`
 - povinné přepínače pro odevzdání úloh
 - *-o jméno* umožní specifikovat vlastní jméno pro přeložený program (namísto `a.out`)
 - `gcc -std=c99 -pedantic -Wall -Wextra -Werror -o hello hello.c`
 - dodatečné doporučené přepínače

dodatečné varování

varování interpretovat
jako error

Hello World (na Aise) – Pokus 2 (pokr.)



```
aisa.fi.muni.cz - PuTTY
/home/xsvenda/Vyuka/pb071/hello: joe hello.c
Processing '/etc/joe/joerc'...Processing '/etc/joe/ftyperc'...done
done
^K I A hello.c (c)  }      Row 12   Col 1   1:21   Ctrl-K H for help
#include <stdio.h>

/*
   This is (possibly) multi
   line commentary
*/
int main() {
    // This is single line commentary
    printf("Hello world\n");
    return 0;
}

File hello.c not changed so no update needed
/home/xsvenda/Vyuka/pb071/hello: gcc -std=c99 -pedantic -Wall -o hello hello.c
/home/xsvenda/Vyuka/pb071/hello: ./hello
Hello world
/home/xsvenda/Vyuka/pb071/hello: █
```

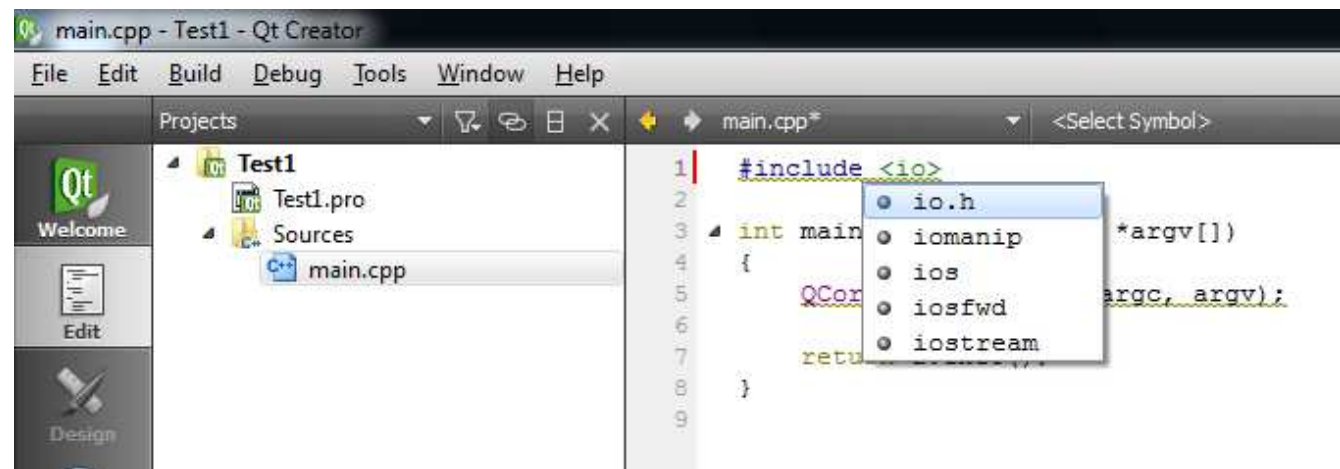
27

Historie, normy
Oblasti použití
Začínáme s C
Nástroje
Lehký průlet C



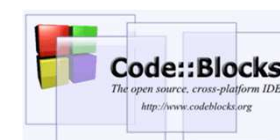
Editor

- Samostatný program (vim, nano, pico, joe...)
- Nebo integrovaný v IDE
 - všechny mají
 - zvýraznění syntaxe, lokalizace chyb, kontextová nápověda...
 - např. QT



Integrated Development Environment (IDE)

- Integrovaný soubor nástrojů pro podporu vývoje
 - typicky s grafickým rozhraním
 - Code::Blocks, Eclipse, Netbeans, Visual Studio, QT Creator, Dev-C++ a mnoho dalších
- Obsahuje typicky:
 - Způsob vytváření a kompilace celých projektů
 - Editor se zvýrazňováním syntaxe
 - WISIWIG GUI editor
 - Pokročilý debugger
 - Profilační a optimalizační nástroje
 - Podporu týmové spolupráce...



Kompilace Aisa

- GNU GCC
 - přepínače (-c, -g, -Wall, -Wextra, -o ...)
 - <http://gcc.gnu.org/onlinedocs/gcc-4.5.1/gcc/Option-Summary.html>
- Překlad přímo do výsledné binárky
 - gcc -std=c99 -pedantic -Wall -o hello hello.c
 - (mezivýsledky jsou smazány)
- Spuštění programu
 - ./hello



The screenshot shows a PuTTY terminal window titled 'anxur.fi.muni.cz - PuTTY'. The terminal output is as follows:

```
/home/xsvenda/Vyuka/pb071/hello: gcc -std=c99 -pedantic -Wall -o hello hello2.c
/home/xsvenda/Vyuka/pb071/hello: ./hello
Hello World
/home/xsvenda/Vyuka/pb071/hello: 
```

31

Překlad po částech

1. **Preprocessing** "gcc -E hello.c > hello.i"
 - rozvinutí maker, expanze include...
2. **Kompilace** "gcc -s hello.i"
 - syntaktická kontrola kódu, typicky chybová hlášení
3. **Sestavení** "as hello.s -o hello.o"
 - assembly do strojového kódu
4. **Linkování** "gcc hello.o"
 - nahrazení relativních adres absolutními



Při běžném překladu proběhnou všechny kroky automaticky, nemusíme pouštět každý zvlášť

32

Překlad po částech - preprocessing

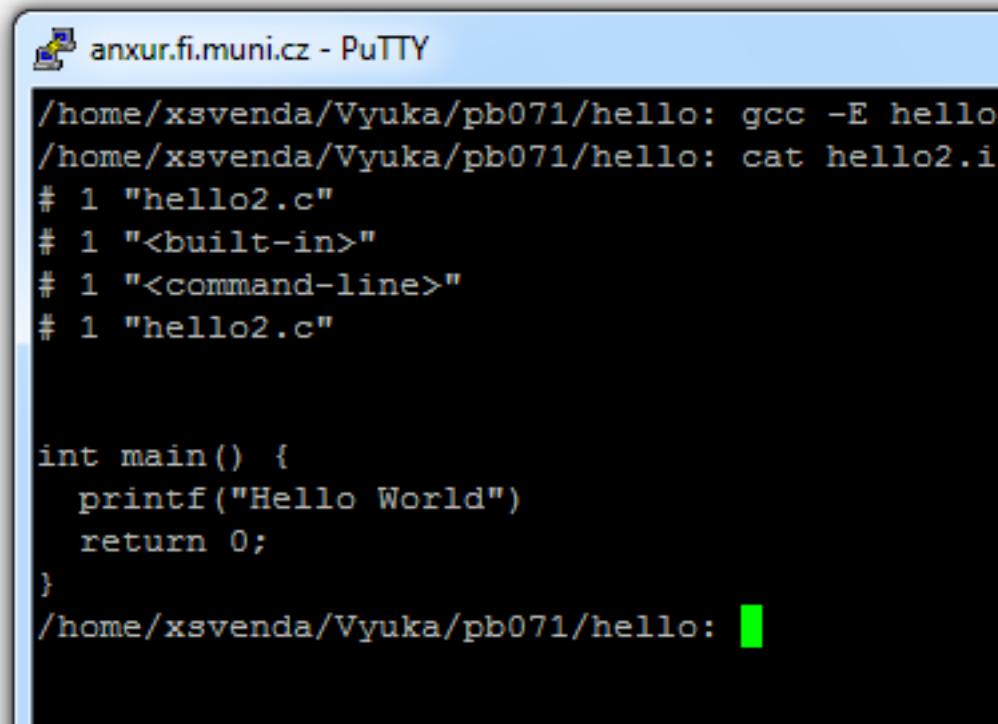
1. Preprocessing "gcc -E hello2.c > hello2.i"

- rozvinutí maker, expanze #include
- odstranění poznámek

hello2.c

```
#define PRINT_MESSAGE "Hello World"

int main() {
    // print on stdout
    printf(PRINT_MESSAGE)
    return 0;
}
```



The screenshot shows a PuTTY terminal window titled "anxur.fi.muni.cz - PuTTY". The terminal displays the output of the command "gcc -E hello2.c". The output shows the preprocessed code, including the expansion of the #define macro and the removal of comments. The preprocessed code is as follows:

```
/home/xsvenda/Vyuka/pb071/hello: gcc -E hello2.c
/home/xsvenda/Vyuka/pb071/hello: cat hello2.i
# 1 "hello2.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "hello2.c"

int main() {
    printf("Hello World")
    return 0;
}
/home/xsvenda/Vyuka/pb071/hello: 
```

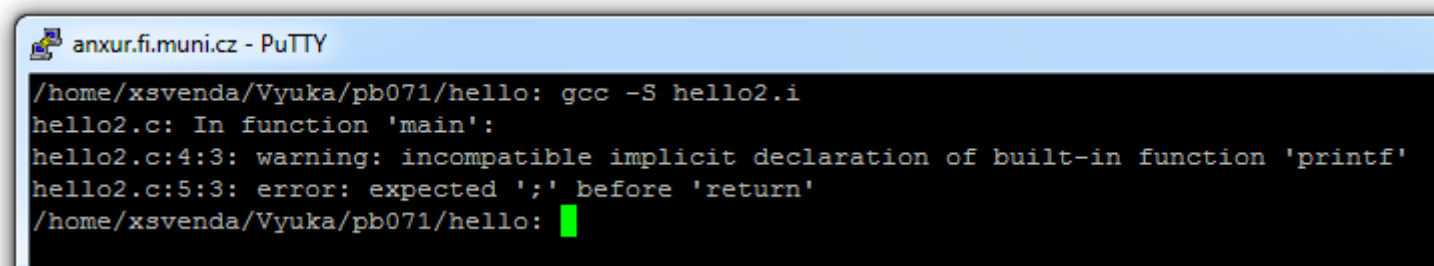
Překlad po částech - **kompilace**

2. Kompilace "gcc -s hello2.i"

- překlad do assembleru, syntaktická kontrola kódu
- zde nastává většina chybových hlášení a varování
- vzniká soubor *.s (pokud nejsou chyby)

hello2.i

```
# 1 "hello2.c"
# 1 "<built-in>"
# 1 "<command line>"
# 1 "hello2.c"
int main() {
    printf("Hello World")
    return 0;
}
```



The screenshot shows a terminal window titled "anxur.fi.muni.cz - PuTTY". The command executed is `gcc -S hello2.i`. The output shows the compilation of `hello2.c` into assembly. It includes a warning about an incompatible implicit declaration of the built-in function `printf` and an error about a missing semicolon before the `return` statement. The terminal path is `/home/xsvenda/Vyuka/pb071/hello`.

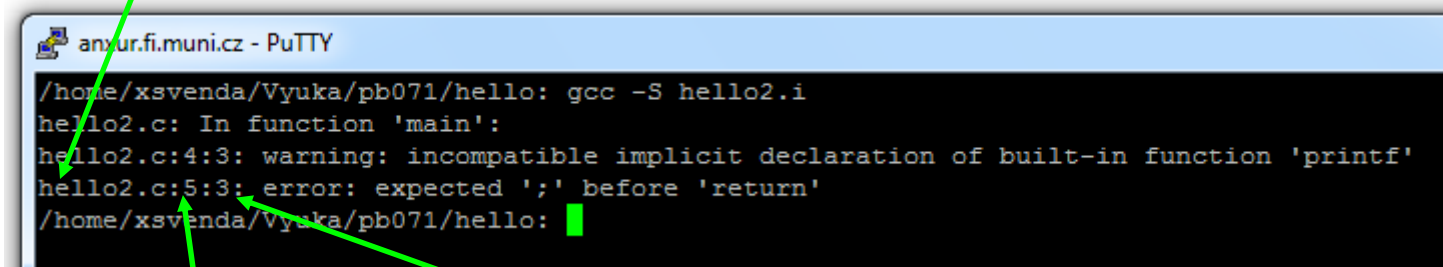
```
anxur.fi.muni.cz - PuTTY
/home/xsvenda/Vyuka/pb071/hello: gcc -S hello2.i
hello2.c: In function 'main':
hello2.c:4:3: warning: incompatible implicit declaration of built-in function 'printf'
hello2.c:5:3: error: expected ';' before 'return'
/home/xsvenda/Vyuka/pb071/hello:
```

34

Jak na chyby (error)?

- Chyby bránící překladu (error)
 - pokud se vyskytnou, nelze program přeložit
 - je nutné v každém případě odstranit
- Začněte s odstraňováním první chyby
 - další mohou být způsobené tou první

soubor obsahující
chybu **[hello2.c]**



```
anyur.fi.muni.cz - PuTTY
/home/xsvenda/Vyuka/pb071/hello: gcc -S hello2.i
hello2.c: In function 'main':
hello2.c:4:3: warning: incompatible implicit declaration of built-in function 'printf'
hello2.c:5:3: error: expected ';' before 'return'
/home/xsvenda/Vyuka/pb071/hello: █
```

řádek s chybou (v
původním *.c souboru) **[5]**

sloupec s chybou **[3]**

35

Jak na chyby (error)? (pokr.)

- Porozumějte chybové hlášce
 - *error: expected ';' before 'return'*
 - v jednoduchých úvozovkách je text z našeho kódu
 - mimo uvozovky je text překladače (popis chyby)
- Google je náš programovací přítel
 - cut&paste chybovou hlášku
- Prozkoumejte i řádek o jedna výše
 - zapomenuté středníky, závorky apod. se detekují až u následujícího příkazu
- Opravte a přeložte znovu

```
#define PRINT_MESSAGE "Hello World"

int main() {
    printf(PRINT_MESSAGE);
    return 0;
}
```

Jak na varování (warning)?

- Varování nebrání překladu programu
 - typicky ale upozorňují na reálný problém
 - může způsobovat problém při sestavení resp. při běhu
- Stejně jako u erroru máte soubor i řádek varování
 - vysvětlení hledejte přes Google
- **Pravidlo 1: vždy kompilujte bez warnings**
 - pokud se zobrazuje 100 varování, nevšimnete si 101
 - budou se vám lépe hledat errorry ve výpisu
- Přepínač překladače –Werror
 - mění varování na error, program se nepřeloží
 - ztrácíte ale rozlišení varování vs. error

37

Jak na varování (warning)? (pokr.)

- warning: incompatible implicit declaration of built-in function 'printf'
 - implicitní deklarace je použití proměnné/funkce bez toho, aby překladač věděl, co je to za funkci
 - printf je funkce, která zde není deklarována
 - google printf → #include <stdio.h>



c printf

About 4,000,000 results (0.05 seconds)

Everything

▶ [printf - C++ Reference](#)

```
#include <stdio.h>
```

```
#define PRINT_MESSAGE "Hello World"
```

```
int main() {  
    printf(PRINT_MESSAGE);  
    return 0;  
}
```

example */ #include <stdio.h> int main() { printf ("Characters: %c %c \n" , 'a' , 65);
"Decimals: %d %ld\n" , 1977, 650000L); printf ...

[plusplus.com/reference/clibrary/cstdio/printf/](#) - Cached - Similar

[Tutorial – printf, Format Specifiers, Format Conversions and ...](#)

printf function is not part of the C language, because there is no input or output There
recently 29 responses to "C Tutorial – printf, ...

[codingunit.com/printf-format-specifiers-format-conversions-and-formatted-output](#) - Cache

[- Wikipedia, the free encyclopedia](#)

... radianic **printf** has its origins in BCPL's writef function. ... also has the **printf** function, with
the same specifications and usage as that in C/C++. ...
[en.wikipedia.org/wiki/Printf](#) - Cached - Similar

Překlad po částech – sestavení

3. Sestavení "as hello2.s -o hello2.o"

- assembly do strojového kódu
- zatím ještě relativní adresy funkcí apod.

hello2.s

```
anxur.fi.muni.cz - PuTTY
/home/xsvenda/Vyuka/pb071/hello: cat hello2.s
.file "hello2.c"
.section .rodata
.LC0:
.string "Hello World"
.text
.globl main
.type main, @function
main:
.LFB2:
pushq %rbp
.LCFI0:
movq %rsp, %rbp
.LCFI1:
movl $.LC0, %edi
movl $0, %eax
call printf
movl $0, %eax
leave
ret
.LFE2:
.size main, .-main
.section .eh_frame,"a",@progbits
.Lframe1:
```

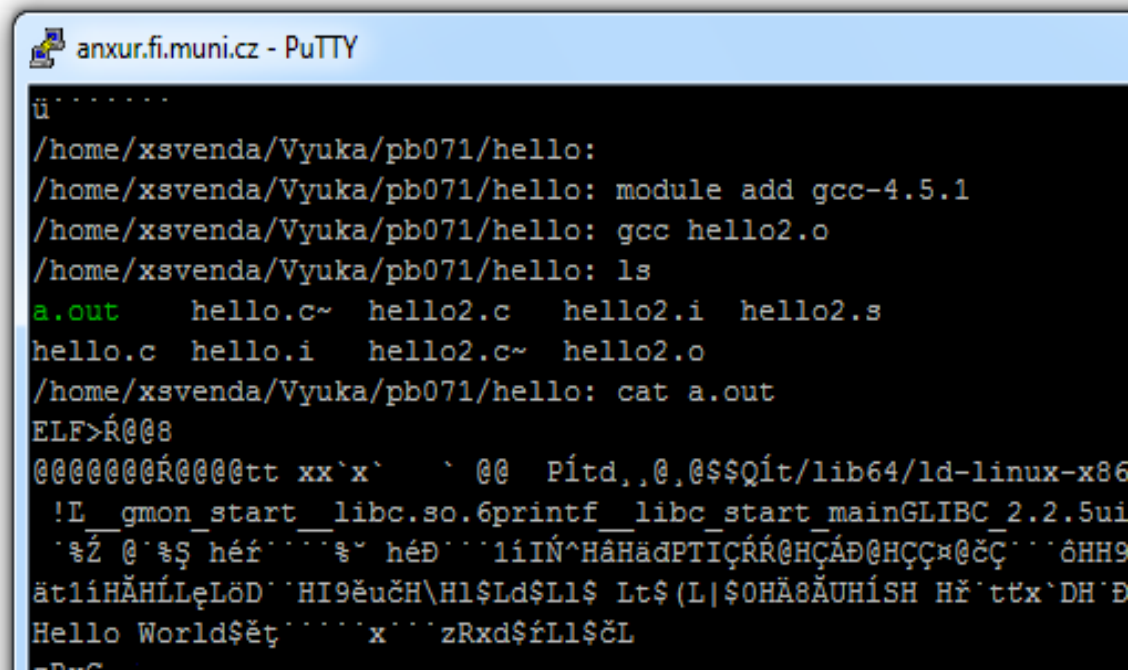
hello2.o

```
anxur.fi.muni.cz - PuTTY
/home/xsvenda/Vyuka/pb071/hello: as hello2.s -o hello2.o
/home/xsvenda/Vyuka/pb071/hello: cat hello2.o
ELF>000
UHíz,č,ĚÅHello WorldzRx
CC: (GNU) 4.1.2 20080704 (Red Hat 4.1.2-50).symtab.strtab.shstrtab
ta.bss.rodata.rela.eh_frame.comment.note.GNU-stack @0
&\1\
>h9R
H .Qîîap
xâ
hello2.cmainprintf
ü.....
/home/xsvenda/Vyuka/pb071/hello: PuTTYPuTTYPuTTYPuTTYPuTTYPuTTY
```

Překlad po částech – linkování

4. Linkování "gcc hello2.o"

- nahrazení relativních adres absolutními
- odstranění přebytečných textů apod.
- objevují se chyby linkování
 - např. chybějící slíbená implementace funkce
- získáme spustitelný program (možnost parametru `-o jméno`)



```
anxur.fi.muni.cz - PuTTY
ü .....
/home/xsvenda/Vyuka/pb071/hello:
/home/xsvenda/Vyuka/pb071/hello: module add gcc-4.5.1
/home/xsvenda/Vyuka/pb071/hello: gcc hello2.o
/home/xsvenda/Vyuka/pb071/hello: ls
a.out  hello.c~  hello2.c  hello2.i  hello2.s
hello.c  hello.i  hello2.c~  hello2.o
/home/xsvenda/Vyuka/pb071/hello: cat a.out
ELF>R@@@
@@@@@@@@R@@@@tt xx`x`  `@@ Pítd,,@,$$Qít/lib64/ld-linux-x86
!E_gmon_start__libc.so.6printf__libc_start_mainGLIBC_2.2.5ui
`$Z @`$ $ hér`'`$~ héd`'1iIN^HÁHädPTIÇRR@HÇÁD@HÇÇx@çÇ`'ôHH9
ät1iHÄHLLeLÖD`HI9ëučH\H1$Ld$L1$ Lt$(L|$OHÄ8ÄUHÍSH Hř`tt`x`DH`E
Hello World$ët`'`x`'zRxd$flL$çL
zRxC
```


QT Creator 2.6.2



- IDE spustitelné na běžných OS (Windows, Linux, MacOS)
- Budeme využívat jako defaultní IDE
 - pokud ale ovládáte dobře jiné, klidně jej použijte
 - oproti Code::Blocks má příjemnější ovládání a lepší ladění
- POZOR: QT není jen IDE, ale i celé API
 - pro zajištění přenositelnosti nestandardizovaných operací poskytuje mezivrstvu QT API (Qxxx objekty)
 - (přenositelnost zdrojového, nikoli spustitelného kódu)
- QT API nebudeme využívat
 - budeme psát a překládat v čistém C
- Tutoriál na <http://cecko.eu/public/qtcreator>
- Stahujte na <http://qt-project.org/downloads>
 - open-source verze (komerční varianta na <http://qt.digia.com>)

41

Doxygen



- Nástroj obdobný jako JavaDoc pro Javu
 - umožňuje generovat dokumentaci z poznámek přímo v kódu
 - speciální formát poznámek (více typů)
- Odevzdávané domácí úkoly musí dokumentaci obsahovat
- Tutoriál na <http://cecko.eu/public/doxygen>

```
/**
 * Display sizes of basic data types
 *
 * @param arraySize    size of dynamically allocated array
 * @return             nothing
 */
void demoDataSizes(int arraySize) {
    #define          ARRAY_SIZE    100
    char    array[ARRAY_SIZE];    // Fixed size array

    ...
}
```

42

Verzovací nástroje



- Nástroj pro verzování kódu a podporu spolupráce v týmu
 - např. SVN, GIT, Mercury...
- V repozitáři (na „serveru“) jsou udržovány všechny provedené změny
 - lze se vrátet zpět na funkční verzi (záloha!)
 - vytvářet oddělené větve...
 - kód z repozitáře by měl jít vždy kompilovat
- Checkout, Update, Commit, dokumentace verzí
- Lze vytvářet vlastní repozitáře
 - např. fakultní SVN, BitBucket...
 - nebo vlastní server (např. VisualSVN Server)
- Domácí úkoly budou odevzdávány přes SVN
- Tutoriál na <http://cecko.eu/public/svn>

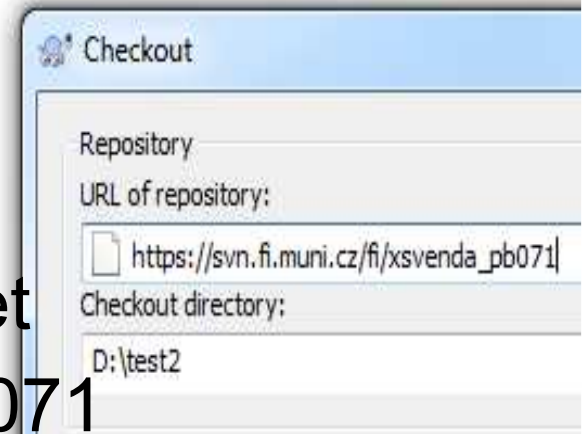
43

Výhody použití verzovacího nástroje

- Při používání jednotlivcem
 1. Záloha práce mimo svůj počítač
 2. Práce na více počítačích (*update*, na konci *commit*)
 3. Návrat zpět na starší verzi (která fungovala)
- Při používání ve skupině
 1. Souběžné práce nad stejnými zdrojáky
 2. Práce vždy nad aktuálními zdrojáky
 3. Možnost práce “offline”
 4. Vytváření nezávislých vývojových větví

Využití fakultního SVN serveru

- <https://fadmin.fi.muni.cz/auth/>
- Počítačová síť → Subversion účet
- Přidat nový repozitář: `login_pb071`
- RClick → SVN Checkout
 - `https://svn.fi.muni.cz/fi/login_pb071`
- Zřídte si alespoň jeden repozitář
 - `login_pb071` na odevzdávání příkladů
- Repozitář neodstraňujte, stačí odstranit soubory



Historie, normy
Oblasti použití
Začínáme s C
Nástroje
Lehký průlet C

F2C – demo (K&R)

- Převod stupňů Fahrenheita na stupně Celsia
 - celsius = $5 / 9 * (fahr - 32)$;

```
#include <stdio.h>

int main(void) {
    int fahr = 0;           // promenna pro stupne fahrenheitu
    int celsius = 0;        // promenna pro stupne celsia
    int dolni = 0;          // pocatecni mez tabulky
    int horni = 300;        // horni mez
    int krok = 20;          // krok ve stupnich tabulky

    fahr := dolni;
    while (fahr <= horni) {
        celsius = 5 / 9 * (fahr - 32);
        // vypise prevod pro konkretni hodnotu fahrenheitu
        printf("%d \t %d \n", fahr, celsius);
        fahr = fahr + krok;
    }
    return 0;
}
```

Nelze zkompilovat a další problémy

47

F2C – demo (problémy)

1. Problém s překladem
2. Problém s celočíselným dělením, implicitní konverze datových typů
3. Výpis proměnných na více číslic
4. Konstanty jako reálná čísla
5. Proměnné jako reálná čísla
6. Výpis proměnných na více desetinných míst
7. Využití příkazu for
8. Symbolické konstanty
9. Samostatná funkce na výpočet převodu

48

F2C – demo (upraveno)

```
#include <stdio.h>
#define F2C_RATIO (5.0 / 9.0)

// samostatná funkce pro vypocet prevodu
float f2c(float fahr) {
    return F2C_RATIO * (fahr - 32);
}

int main(void) {
    int fahr = 0;           // promenna pro stupne fahrenheitu
    float celsius = 0;      // promenna pro stupne celsia
    int dolni = 0;          // pocatecni mez tabulky
    int horni = 300;        // horni mez
    int krok = 20;          // krok ve stupnich tabulky

    for (fahr = dolni; fahr <= horni; fahr += krok) {
        celsius = f2c(fahr);
        // vypise prevod pro konkretni hodnotu fahrenheitu
        printf("%3d \t %6.2f \n", fahr, celsius);
    }
    return 0;
}
```

49

F2C – se vstupem od uživatele

```
#include <stdio.h>
#define F2C_RATIO (5.0 / 9.0)
int main(void) {
    int fahr = 0;           // promenna pro stupne fahrenheitu
    float celsius = 0;      // promenna pro stupne celsia
    int dolni = 0;          // pocatecni mez tabulky
    int horni = 300;        // horni mez
    int krok = 20;          // krok ve stupnich tabulky

    // vypiseme vyzvu na standardni vystup
    printf("Zadejte pocatecni hodnotu: ");
    // precteme jedno cele cislo ze standardniho vstupu
    scanf("%d", &dolni);

    for (fahr = dolni; fahr <= horni; fahr += krok) {
        celsius = F2C_RATIO * (fahr - 32);
        // vypise prevod pro konkretni hodnotu fahrenheitu
        printf("%3d \t %6.2f \n", fahr, celsius);
    }
    return 0;
}
```

50

Co si můžete hned doma vyzkoušet

- Připojte se na Aisu a zkuste kompilaci s gcc
- Nainstalujte QT Creator, zkuste si vytvořit projekt
- Pohrajte si s SVN
 - založte si SVN repozitář na fadmin.fi.muni.cz
 - nainstalujte si SVN klienta, zkuste Update/Commit
- Nalad'te si <http://www.se-radio.net/> 😊

Shrnutí

- Organizační – vše na <http://cecko.eu/public/pb071>
- Hlavním cílem předmětu je trochu programovat
- Používejte nástroje (default QT Creator, SVN)
- Domácí úkoly zadávány/probírány na cvičeních
 - http://cecko.eu/public/pb071_cviceni
 - odevzdání na Aise
- Nebojte se zeptat!
 - přednáška, cvičení, poradci, konzultačky...

