

PB071 – Principy nízkoúrovňového programování

Úvod, organizace, nástroje

IS, 1998



2021



Slidy pro komentáře (děkuji!):

<https://drive.google.com/file/d/12BiGVNc4yYCq9WRZkXGJg4MPIkSeatSp/view?usp=sharing>

1

Cíle předmětu

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

Top questions (1)



Anonymous

0 👍

Pokud bude program přeložen na Linuxu, bude jej možné spustit v OS Windows?

- Pokud budete mít během poslechu přednášky dotaz, tak jej vložte na [slido.com \(#pb071_2024\)](https://slido.com/#pb071_2024)
- Společně projedeme dotazy každé pondělí během přednášky Q&A

Join at
slido.com
#pb071_2024

Organizační

Historie, normy

Oblasti použití

Začínáme s C

Nástroje

Lehký průlet C





Principy nízkoúrovňového programování

Vítejte na stránkách předmětu *PB071 Principy nízkoúrovňového programování*.

Cíle předmětu

Cílem **PB071** je naučit studenty základní syntaxi jazyka C podle norem ANSI a ISO/IEC, dekomponovat problémy, používat moderní vývojové nástroje a ovládat dobré programátorské návyky. Důraz je kladen na pochopení paměťového modelu, správu paměti a korektní používání dynamické alokace.

Důležité odkazy

 Organizační pokyny	důležité
 Aktuality na fóru	IS
 Materiály k přednáškám a cvičením	
 Domácí úkoly	žádné

Konzultace

- nebojte se konzultovat problémy s domácími úkoly nebo s probíranou látkou
- využijte primárně konzultace přímo na cvičení nebo u studentských poradců

Semestr Jaro 2024 je prezenční (a poslední)

- V předstihu v pátek dostupné slidy pro přednášku
 - Projděte si dopředu, zapište otázky na slido
- Přednáška je standardní prezenční
 - Vybrané témata + Kahoot + Q&A + bonus
- Cvičení standardní prezenční v rámci vaší seminární skupiny
- Poslední běh předmětu

Poslední běh předmětu PB071

- Jaro 2024 je poslední běh předmětu PB071
 - Podmínky pro absolvování předmětu podobné jako v minulosti
 - Nahrazeno předmětem PB111 Principy nízkoúrovňového programování (P. Ročkai, A. Matoušek)
 - <https://is.muni.cz/auth/predmet/fi/jaro2024/PB111>
 - V případě neúspěchu v tomto roce opakujete PB111 (Jaro 2025)
- Nová verze “PB071” se zaměřením na pokročilejší C poběží zřejmě od podzimu 2025
 - Předmětová alternativa k Java, C++, C#...
 - Bude mít jiný kód, není přímá náhrada za PB071

Organizační (1)

- Jazyk: C, občas česky a slovensky
- Přednášky
 - nepovinné, ale snad přínosné a zábavné 😊 (prezenční)
 - Přednášky přednahrané (IS), bodované odpovědníky
 - Předpoklad základní znalosti algoritmizace (co je cyklus...)
 - Rozcestník <https://www.fi.muni.cz/pb071>
- Cvičení
 - povinné, dvouhodinové, dvě neúčasti tolerovány (sem. skupiny)
 - Předpoklad účasti na vaší skupině, po domluvě možnost využití jiné
 - aktivní práce na příkladech a domácích úkolech, konzultace
 - podklady <https://www.fi.muni.cz/pb071/seminars/>
- Ukončení předmětu
 - **Prostudujte** <https://www.fi.muni.cz/pb071/info/>
 - Úspěšné zakončení předmětu (podmínky + minimum bodů)
 - **semestrové povinnosti** (domácí úkoly + testíky, zisk minima bodů + 4 nenulové kladné domácí úkoly)
 - **úspěšná programovací zkouška** (krátký test + programovací příklad na PC)
 - **teoretická část zkoušky** (teoretický test na PC)

Organizační (2)

- Slidy z přednášky, ukázkové zdrojáky
 - <https://www.fi.muni.cz/pb071>
- Domácí úkoly
 - 5+1 za semestr, zadávány průběžně (na webu cvičení)
 - body za funkčnost, body za správné a včasné odevzdání
 - deadline pro odevzdání (na stránce úkolu, 2 týdny, 24:00)
 - budou zveřejňována ukázková řešení
- Odevzdání/testování
 - možnost odevzdání **nanečisto** (částečně omezeno, detaily na cvičení)
 - odevzdání do fakultního git, spuštění notifikačního skriptu
 - při odevzdání alespoň týden dopředu získáváte jeden pokus naostro navíc
- Obtížnost úkolů se postupně zvyšuje
 - Nenechávejte si řešení na poslední chvíli
 - První domácí úkoly jsou typicky snažší body!

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
 - s řešeními z minulých let (pokud je podobný příklad)
 - u podezřelých příkladů probíhá manuální kontrola
- V případě opsání jsou potrestáni oba účastníci

Neopisujte



● Co je OK

- Společně diskutujete ústně možnosti řešení
- Společně diskutujete včetně využití tabule, z tabule ale neopisujete ani si ji nefotíte (pochopení v hlavě)
- Necháte si poradit od konzultanta

● Co není OK

- Pošlete svůj kód kolegovi
- Najdete kód na internetu a zkopírujete ho (Ctrl+C, přepis)
- Napíšete kus kódu kolegovi nebo diktujete co má psát
- Nechat si napsat od někoho/něčeho jiného
- ... (pokud si nejste jisti, zeptejte se nás)

ChatGPT a spol

- Velký příslib a překvapivě dobrá funkčnost
 - Pozor, dělá občas opravdu chyby
- Jednodušší úlohy nejspíš zvládne
 - Zvládnete ale vy ty obtížnější, když si necháte udělat ty jednodušší od AI?
 - Obecný problém pro styl současné výuky (nutné změny)
- Pravidla použití stejná jako pro kód z internetu
 - Můžete se inspirovat, ale nesmíte kód kopírovat
 - Odevzdaný úkol musí být vaše programování
- Umožňuje okomentovat funkčnost vašeho kódu

Kontakt

IS, 1998



2021



- Přednášející
 - Petr Švenda, svenda@fi.muni.cz
 - konzultační hodiny: přednášky, pátky 10:00 – 11:00 A406
 - dotazy k přednášce, celkové organizaci, známky, eskalace problémů
- Cvičící
 - na hodinách – váš primární kontakt, využijte hojně
- Studentští konzultanti
 - dodatečné konzultace nezávisle na skupině
 - v definované konzultační hodiny (viz hlavní web)
 - od druhého týdne semestru

Sběr zpětné vazby



- Občasné dotazníčky (obtížnost úloh, porozumění...)
- Samozřejmě možné osobně
- Předmětová anketa
 - Piště prosím témata/věci/postřehy, které se Vám líbila i nelíbila
 - Anketu čteme a předmět podle ní vylepšujeme, má to cenu!
- Velké poděkování všem předem za poznámky a náměty!

Komentáře k slidům z přednášky

- Není nutné se přihlašovat, jakékoli poznámky ke slidům
- Faktické chyby, příliš komplikované (probírat déle), příliš snadné (probírat rychleji), návrh na rozšíření...
- Link na komentovatelné slidy uveden vždy na prvním slidu dané přednášky

The image shows a presentation slide on the left and a comment overlay on the right. The slide is titled 'PB071_01_UvodOrganizaceNastroje_2019.pptx' and lists several topics: 'Organizační', 'Historie, normy', 'Oblasti použití', 'Začínáme s C', 'Nástroje', and 'Lehký průlet C'. The 'Oblasti použití' item is highlighted with a yellow bar. The comment overlay is from an anonymous user, dated '10:03 Dnes', and contains the text: 'Není mi jasné X...', 'Překlep Y...', and 'Zbytečně dlouho probíráno Z...'. There is a 'Vyřešit' button next to the comment.



Kolektiv PB071 2010-2024

Přímé zapojení: 151

Včetně testerů: >> 220


Zpětná vazba: >> 2400


Předpoklady, návaznost na další předměty


- Předpoklady
 - předchozí zkušenost s libovolným programovacím jazykem (vlastní nebo IB111)
 - základy algoritmizace
 - (příkazy, podmínky, cykly, funkce, koncept proměnné)
- Na předmět PB071 navazuje
 - PB161 Programování v jazyce C++ (4. semestr)
 - PB162 Programování v jazyce Java (4. semestr)
 - PB173 Tématické programování (3,4,5. sem.)
 - práce v laboratořích (nebojte se zeptat)

<https://www.fi.muni.cz/pb071/tutorials/>


 [Hlavní stránka](#)


 [Informace k rektorskému volnu](#)

 [Pokyny](#) ▾

 [Podklady k přednáškám](#)



 [Cvičení](#) ▾


 [Domácí úkoly](#) ▾

 [Miniúkoly](#)


 [Manuál](#)


 [Tutoriály \(starší verze\)](#) ▾

 [Autentizovaná sekce pro cvičící](#) 

 [Začínáme](#)


▸ [Rozdiely oproti jazyku Python](#)

 [Kde hľadať dokumentáciu jazyka](#)

 [Aisa a kontr, pripojenie pomocou ssh](#)

▸ [Moduly](#)

▸ [\(voliteľné\) SSH kľúče a automatické prihlasovanie](#)


 [Vývoj na Linuxe](#)

▸ [Možnosti](#)

▸ [Shell a bash](#)

▸ [Terminálové nástroje](#)

▸ [Terminal multiplexing](#)

 [Vývoj na Windows](#)


▸ [Možnosti](#)

▸ [Pripojenie na fakultný server](#)

▸ [Windows Subsystem for Linux \(WSL\)](#)


▸ [Virtualizácia](#)

▸ [Inštalácia nástrojov](#)

 [Vývoj na macOS](#)

▸ [Homebrew a potrebné nástroje](#)

▸ [Xcode](#)

 [IDE \(vývojové prostredie\)](#)

▸ [Qt Creator](#)

▸ [CLion](#)

C vs. Python

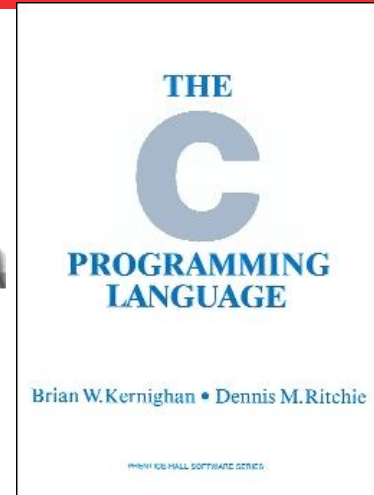
- Připravili jsme pro vás přehled základních rozdílů
 - díky Marku Klučárovi!
 - <https://www.fi.muni.cz/pb071/man/c-vs-python.html>
- Práce na Unixových strojích
 - Požadujeme jen elementární základy (ssh, gcc)
 - <http://www.ee.surrey.ac.uk/Teaching/Unix/>

*“Unix was not designed to stop you from doing stupid things,
because that would also stop you from doing clever things”*

Doug Gwyn

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 (většina) proměnných znám v době překladu

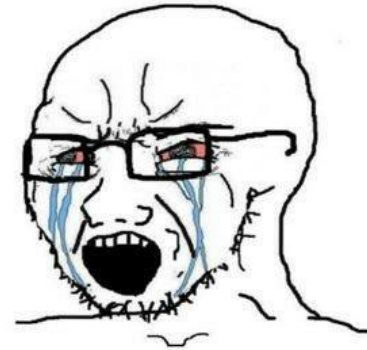
- **Není objektově orientovaný**

- nejsou přímo jazykové konstrukce (např. typ class)

New languages programmers



Python is better

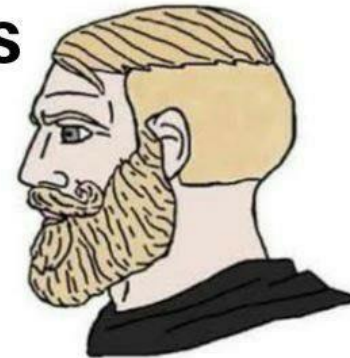


**Nooooo, JS is
far better**



C is hell

C programmers



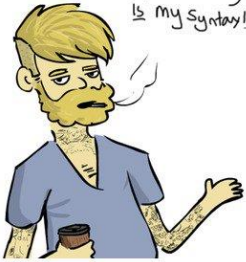
Yes

Proč se učit a používat jazyk C

- Porozumění fundamentům provádění výpočtů
 - Vysokouúrovňové jazyky spoustu důležitých detailů skrývají
- 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
 - <https://benchmarksgame-team.pages.debian.net/benchmarksgame/index.html>

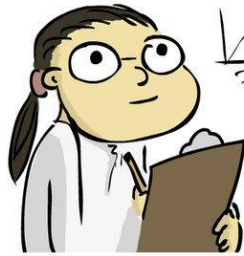
PYTHON

My formatting
is my Syntax!



R

$\frac{L_i}{FCI}$



JAVA



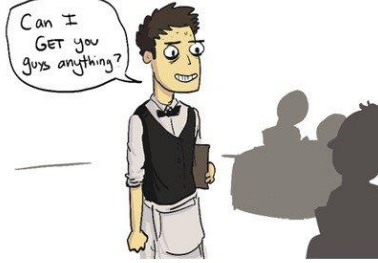
JAVASCRIPT

I'm technically functional.



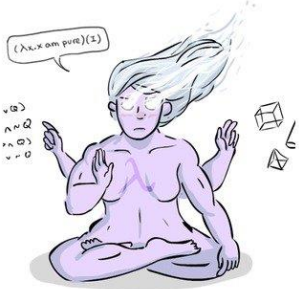
PHP

Can I
GET you
guys anything?



HASKELL

(lambda pure)(2)



PERL



C
If you're not
getting memory,
you're not long!

C++
If you're not
getting memory,
you're not long!

C#

I am
Nothing
like my
father.



RUBY ON RAILS



ASSEMBLY



ERLANG

DESTROY ONE OF MY
PROCESSES & I WILL
ONLY GROW STRONGER



GO



BRAINFUCK



Rust

I'll be designated
driver tonight!
We know
Don't talk while
eating you could
choke and die
haha



http://leftoversalad.com/c/015_programmingpeople/

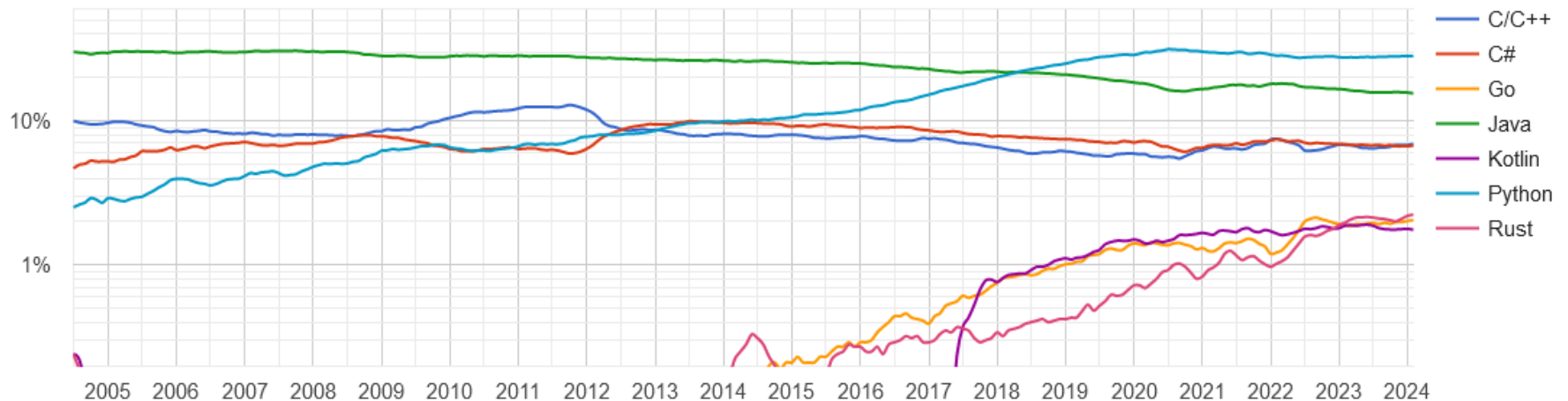
Popularita jazyku

- Zjišťovat popularitu jazyků není snadné
 - Množství řádek kódu? Počet vývojářů? Počet systémů v daném jazyce? Absolutně nebo přírůstek za čas?
- PYPL PopularitY of Programming Language Index
 - Podíl vyhledávání 'X tutorial' dle Google Trends

Worldwide, Feb 2024 :

Rank	Change	Language	Share	1-year trend
1		Python	28.11 %	+0.6 %
2		Java	15.52 %	-1.0 %
3		JavaScript	8.57 %	-1.0 %
4	↑	C/C++	6.92 %	+0.1 %
5	↓	C#	6.73 %	-0.1 %
6	↑	R	4.75 %	+0.7 %
7	↓	PHP	4.57 %	-0.6 %
8		TypeScript	2.78 %	-0.0 %
9		Swift	2.75 %	+0.5 %
10		Objective-C	2.37 %	+0.1 %
11		Rust	2.23 %	+0.3 %
12		Go	2.04 %	+0.1 %
13		Kotlin	1.75 %	-0.1 %
14		Matlab	1.64 %	-0.1 %
15	↑↑	Ada	1.08 %	+0.2 %
16	↑↑	Dart	0.98 %	+0.2 %
17	↓	Perl	0.98 %	-0.0 %
18		Powershell	0.94 %	+0.1 %
19	↓↓↓↓	VBA	0.88 %	-0.1 %

PYPL PopularitY of Programming Language



<https://pypl.github.io/PYPL.html>

The Computer Language Benchmarks Game

“Which programming language is fastest?”

Should we care? How could we know?

“It's important to be realistic: most people don't care
about program performance most of the time.”

- <https://benchmarksgame-team.pages.debian.net/benchmarksgame/which-programs-are-fastest.html>

Benchmarks game

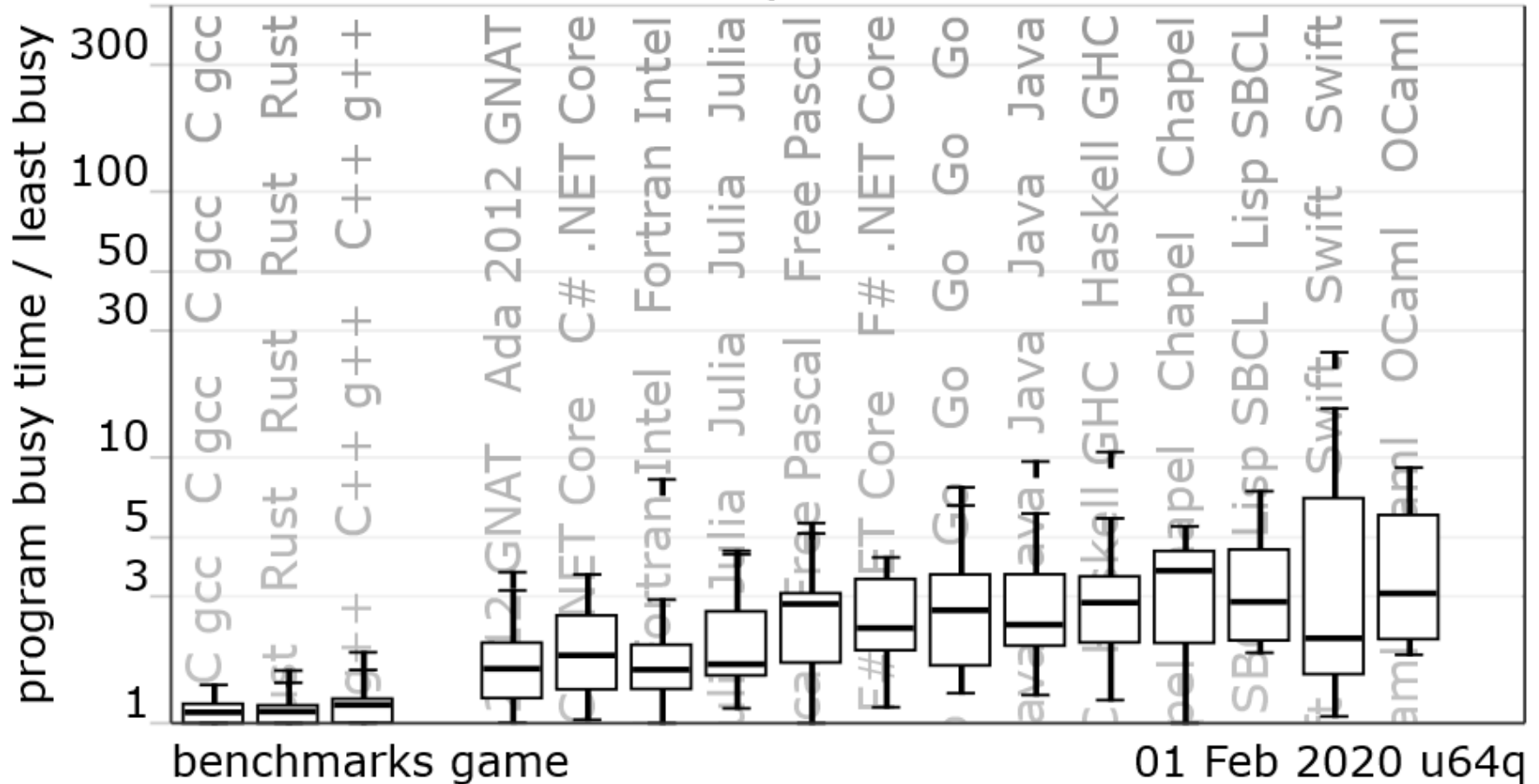
- Srovnávat rychlost jazyků není snadné
 - Konkrétní úloha, cílový hardware (CPU jádra, cache...)
- Idea testování
 - Několik různých, jasně zadaných úkolů
 - Spouštěno na jednotné hardware
 - Programátoři píší svoje co řešení, to nejrychlejší se počítá
- Pozor na vyhodnocení!

Always look at the source code.

Look at the slower simple sequential programs, *and* look at the parallel programs written for multicore, *and* look at the low-level programs written for SIMD.

Celkové výsledky přes různé programy

How many times slower?



Srovnání rychlostí – práce s poli

<https://benchmarksgame-team.pages.debian.net/benchmarksgame/performance/revcomp.html>

× source	secs	mem	gz	cpu secs	cpu load
1.0 C gcc #8	0.48	499,404	1543	0.80	93% 31% 24% 22%
1.7 Rust #2	0.80	995,004	1330	1.45	58% 45% 27% 48%
1.7 <u>Rust #3</u>	0.81	995,004	1097	1.50	59% 46% 27% 50%
1.9 <u>C gcc #6</u>	0.89	715,812	820	1.28	0% 100% 25% 16%
1.9 Chapel #8	0.91	1,056,436	1298	2.70	93% 65% 66% 73%
2.1 <u>C gcc #2</u>	1.01	993,892	750	1.52	86% 30% 19% 13%
2.8 Go #6	1.33	1,278,596	1338	2.46	45% 31% 78% 31%
3.0 Julia #8	1.44	978,216	522	1.71	6% 100% 6% 6%
3.0 <u>C gcc #5</u>	1.46	993,848	647	1.46	0% 0% 99% 0%
3.3 <u>Chapel #3</u>	1.57	1,884,484	764	2.93	94% 29% 29% 34%
3.3 C# .NET #6	1.58	1,026,664	1621	3.78	21% 98% 45% 74%
3.5 <u>Julia #9</u>	1.68	975,488	449	1.95	6% 5% 99% 5%
3.5 <u>Chapel #5</u>	1.69	1,884,480	610	2.99	94% 26% 26% 31%
3.7 <u>Go #2</u>	1.76	814,544	611	1.80	13% 0% 88% 0%
4.0 <u>C# .NET #7</u>	1.92	954,396	1310	5.06	48% 93% 89% 33%
4.1 <u>Go #3</u>	1.96	814,352	605	2.00	12% 3% 86% 0%
4.3 Ada 2012 GNAT #2	2.05	821,756	3283	2.37	40% 0% 0% 76%
4.9 OCaml #3	2.33	11,112	1356	6.06	60% 67% 64% 67%

C program (tento konkrétní) není optimalizovaný pro více jader

3.0 Julia #8	1.44	978,216	522	1.71	6% 100% 6% 6%
3.0 <u>C gcc #5</u>	1.46	993,848	647	1.46	0% 0% 99% 0%
3.3 <u>Chapel #3</u>	1.57	1,884,484	764	2.93	94% 29% 29% 34%

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.

×	Program Source Code	sort CPU secs	sort Elapsed secs	sort Memory KB	sort 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%

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	
x	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**
- Nevhodné pro
 - webové aplikace (PHP, JavaScript...)
 - rychlé prototypy (ale nutno znát dobře jiný jazyk)
 - větší projekty vyžadující objektově orientovaný návrh (C++, Java...)



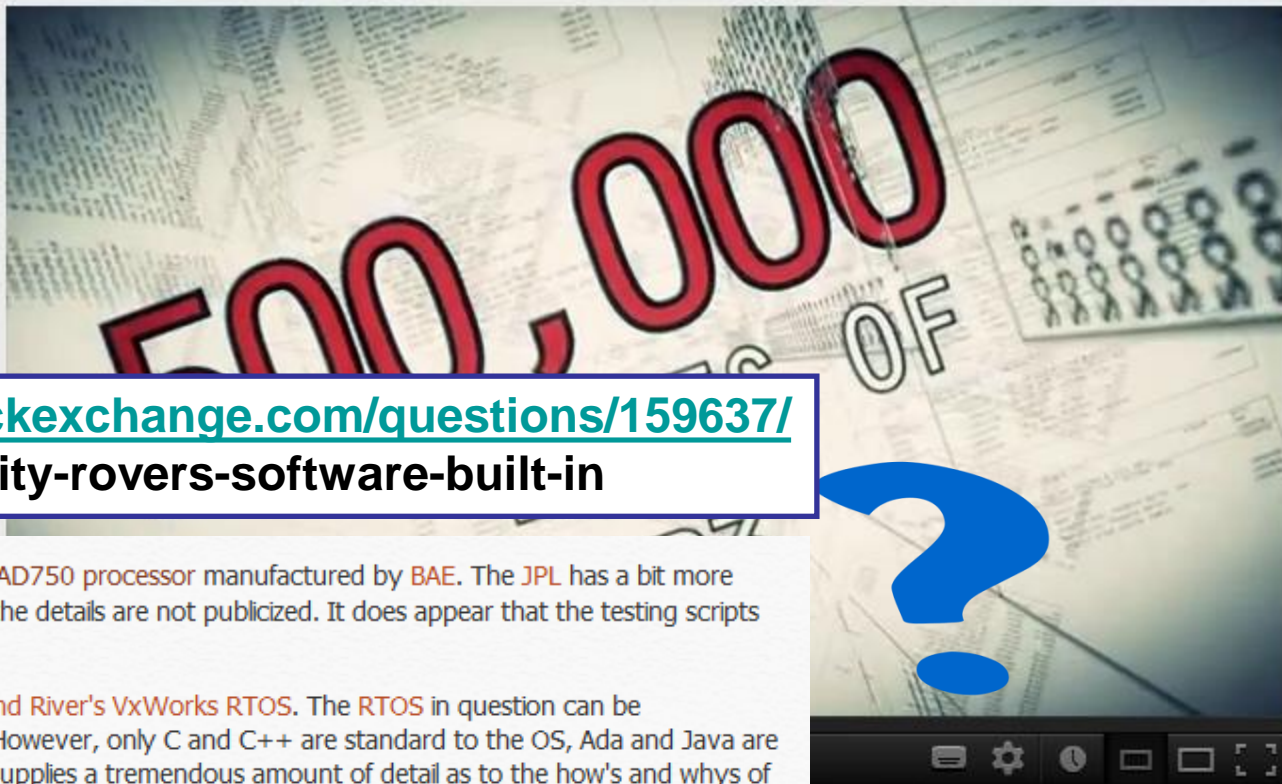
Challenges of Getting to Mars: Curiosity's Seven Minutes of Terror

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

nges of
ars.

33

Normy, standardy a rozšíření

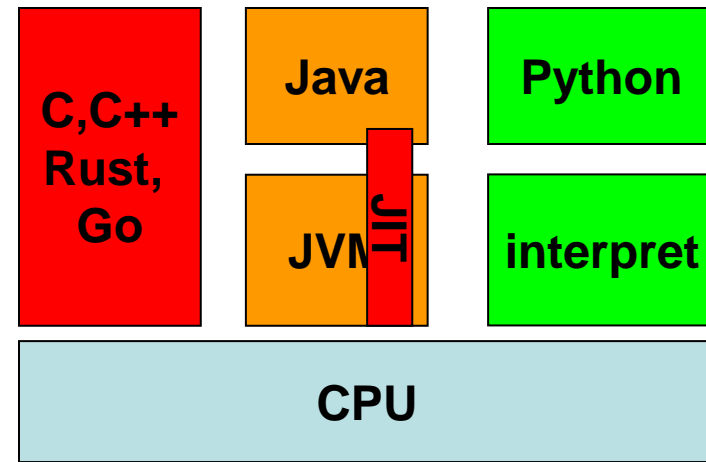
- Kniha The C Programming Language (1978)
 - neformální norma
- ANSI X3.159-1989 (ANSI C, Standard C, C89)
- ISO/IEC 9899:1990 (jen převzaté ANSI C, C90)
- ISO/IEC 9899:1999 (C99)
 - gcc -std=c99
 - (budeme využívat při psaní a domácích úlohách)
- ISO/IEC 9899:2011 (C11, defacto nejnovější)
 - probereme některá rozšíření (vlákna, synchronizace...)
 - ISO/IEC 9899:2018 jen opravy
 - [http://en.wikipedia.org/wiki/C11_\(C_standard_revision\)](http://en.wikipedia.org/wiki/C11_(C_standard_revision))
- (*Finalizován standard C23 – plánovaná publikace 2024*)
 - [https://en.wikipedia.org/wiki/C23_\(C_standard_revision\)](https://en.wikipedia.org/wiki/C23_(C_standard_revision))

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++, Rust, Go...
 - překlad do strojového kódu (Ahead Of Time)
 - překlad nutný zvláště 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 platformem
 - bytecode interpretovaný, ale JIT (Just-In-Time) kompilátor
 - Lze provést i AOT (snížená přenositelnost, ale menší paměťové nároky...)
- Skriptovací imperativní: Perl, Python...
- typicky se interpretuje, platformově nezávislé (pokud je interpret)
- Funcionální: Haskel, LISP...
 - jiné paradigma: matematický zápis odvození z počátečních hodnot
- Logické programování: Prolog...
- jiné paradigma: JAK má výsledek vypadat, ne jak se k němu dostat
- Aktuální trend je slučování různých paradigmat



Jazyk C je vysokoúrovňový! 😊

- “C is a very nice high-level programming language with many of the modern programming constructs in it” <https://youtu.be/tc4ROCJYbm0?t=1149>

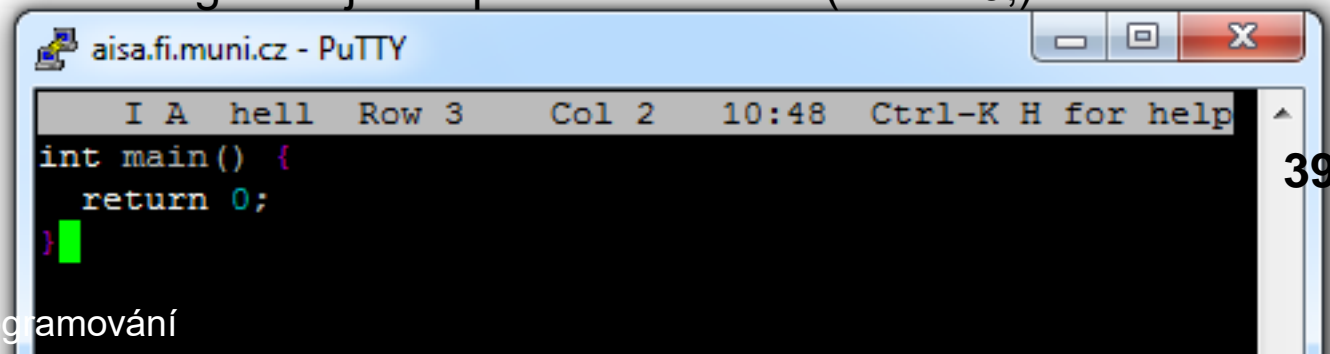


Historie, normy
Oblasti použití
Začínáme s C
Nástroje
Lehký průlet C

Hello World (na Aise) – Pokus 1

1. Připojíme se na Aisu (2x, pro edit & pro překlad)
 - Windows/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 a `void`)
4. Implementujeme tělo funkce `main`
 - do složených závorek `{ }`
 - vrátíme hodnotu signalizující úspěšně ukončení (`return 0;`)
 - uložíme

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



The screenshot shows a PuTTY terminal window titled "aisa.fi.muni.cz - PuTTY". The terminal displays the following C code:

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

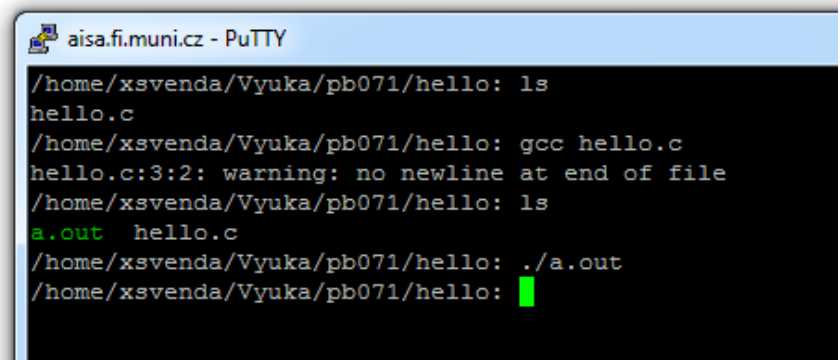
A green cursor is visible at the end of the closing brace of the `main` function.

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

40

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(void) {
  // This is single line comment
  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"

41

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`
 - Jaro 2024: gcc (GCC) 11.4.1
 - OK, není potřeba dělat žádné dodatečné kroky
 - Pokud stará verze, lze nastavit pomocí `module add`
 - `module add gcc-10.2` (přidejte si do `.profile`)

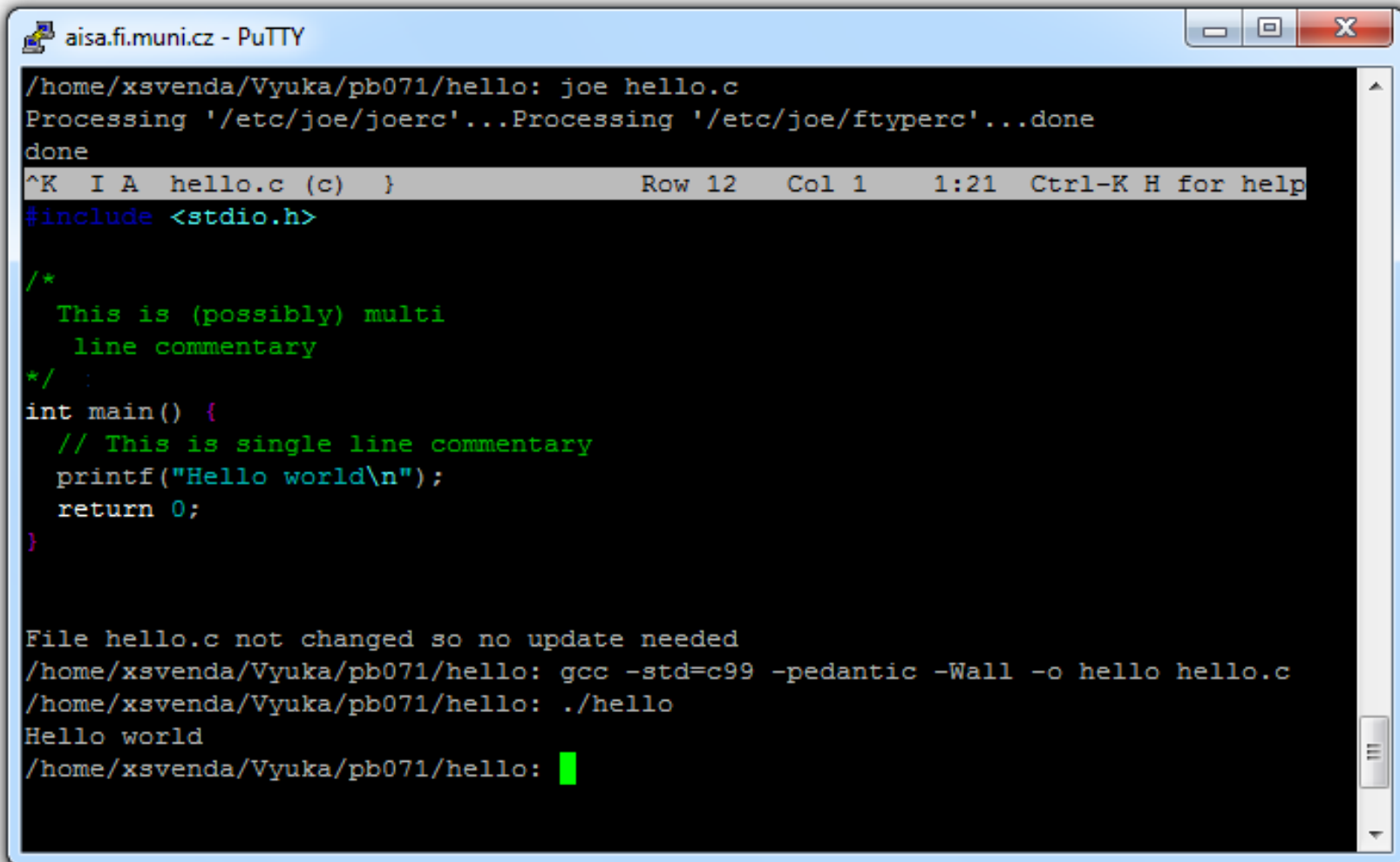
Hello World (na Aise) – Pokus 2 (pokr.)

- 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: █
```

Jak na chyby (error) a varování (warnings)?

```
#define PRINT_MESSAGE "Hello World"

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

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]

```
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: █
```

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

sloupec s chybou [3]

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 v kódu 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

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>

Google

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
currently 29 responses to "C Tutorial – printf, ...

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

[- Wikipedia, the free encyclopedia](#)

... also has the printf function, with
the same specifications and usage as that in C/C++. ...

[en.wikipedia.org/wiki/Printf](#) - Cached - Similar

Kahoot!

Kahoot!

- Připravte si notebook / telefon s připojením na net
 - **wlan_fi** → <https://wifi.fi.muni.cz/> (Aisa login / heslo)
 - **Eduroam** (učo@eduroam.muni.cz / sekundární heslo)
- Pokud nemáte, nevadí (připojte se k sousedovi)
- Budeme používat v průběhu celého semestru

- Navštivte stránku <https://kahoot.it> a jdeme na to!
 - PB071 Prednaska 01 - Úvod C

PB071 Prednaska 01 - Úvod C

Kahoot!

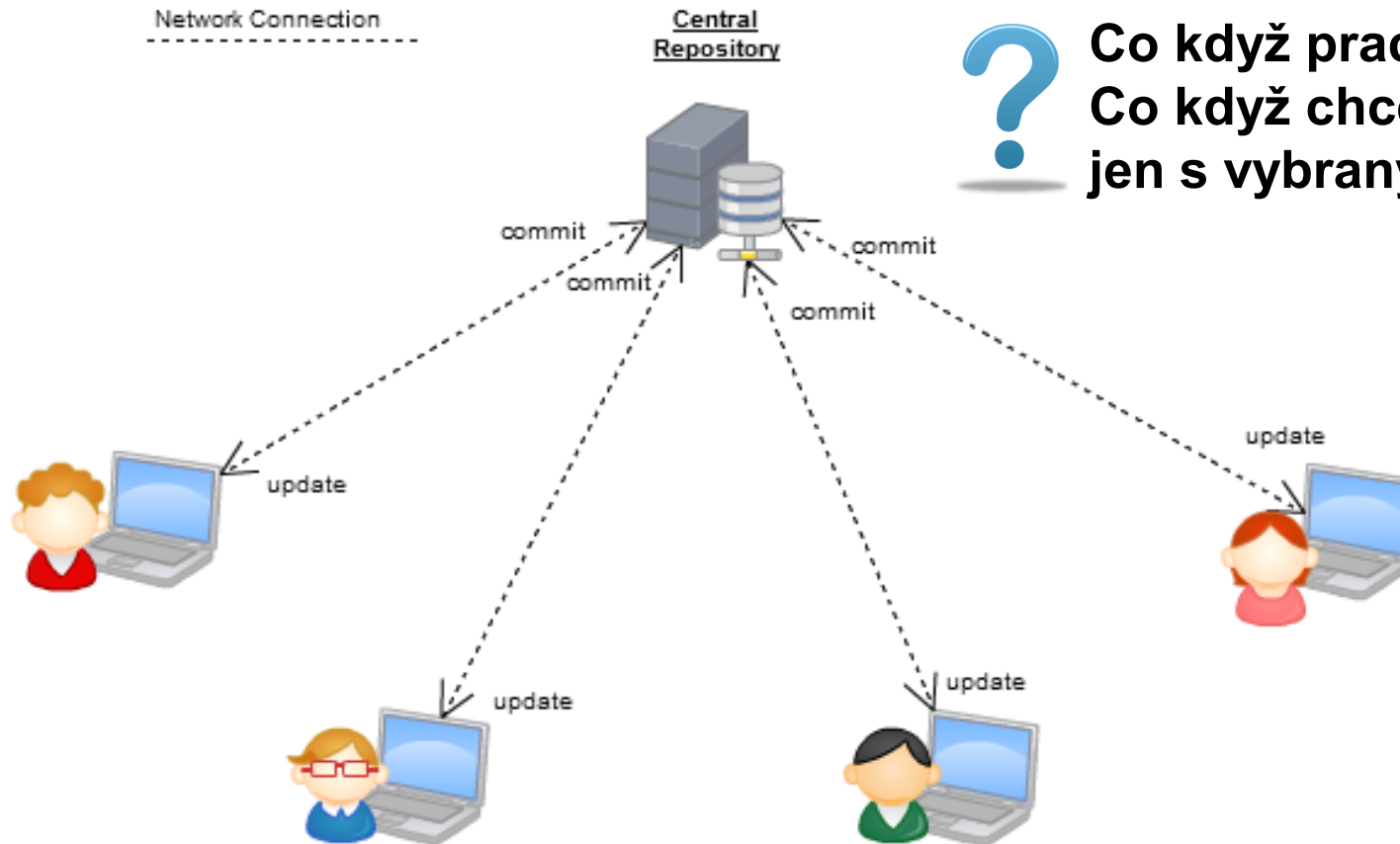


Historie, normy
Oblasti použití
Začínáme s C
Nástroje
Lehký průlet C



VERZOVÁNÍ KÓDU

Centrální repozitář (např. SVN) SUBVERSION



Co když pracujete offline?
Co když chcete spolupracovat
jen s vybraným uživatelem?

Zdroj obrázků: [Joomla's Documentation](#)

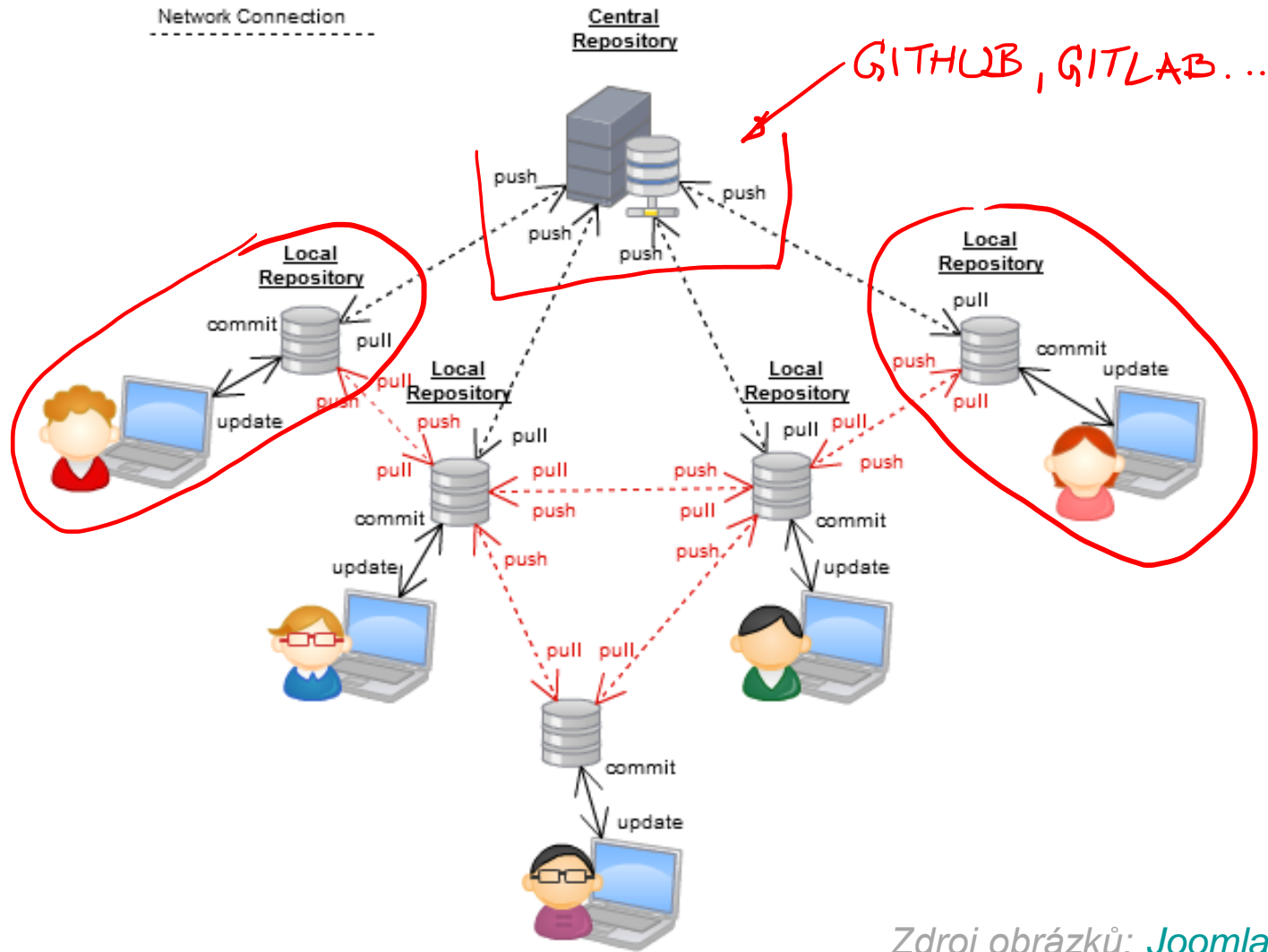
5+1 důvodů proč verzovat (a používat git)

1. Co když si omylem smažu adresář s kódy?
 - Velice inteligentní záloha
2. Co když pracuji ze dvou a více strojů?
 - Inteligentní přenos a synchronizace kódů
3. Co když se potřebuji vrátit ke starší verzi kódů?
 - Proto, že fungovala nebo v ní zákazník našel chybu
4. Co když pracuji s dvěma a více lidmi?
 - Změny probíhají nad stejnou částí kódu
5. Co když spouštím složité analýzy a testy?
 - Kontinuální integrace na serveru přes noc

5+1 důvodů proč verzovat (a používat git)

- Verzování pomocí gitu bude probíráno na cvičení

Lokální a centrální repozitář (např. git)



Zdroj obrázků: [Joomla's Documentation](#)

Využití fakultního GitLab serveru

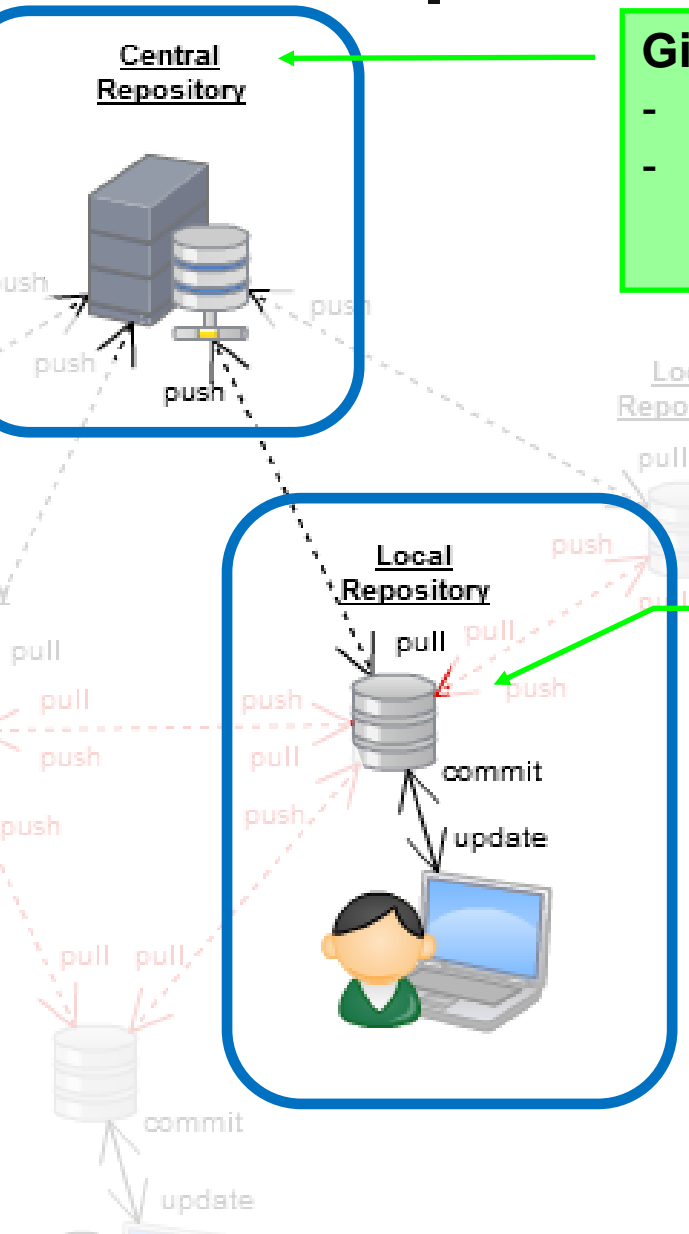


- <https://gitlab.fi.muni.cz/>
- LDAP Username/password
 - Vaše přihlašovací údaje na aisu
- New project → Project name → Create project
- Command line instructions
- Domácí úkoly budou odevzdávány přes Git

```
git clone git@gitlab.fi.muni.cz:xsvenda/hello.git
cd hello
touch README.md
git add README.md
git commit -m "add README"
git push -u origin master
```

60

Situace pro PB071 (git, ale jeden uživatel)



Git repozitář

- Server `gitlab.fi.muni.cz`
- Repo vytvořeno přes webové rozhraní (návod na <https://www.fi.muni.cz/pb071/tutorials/git/>)

Váš počítač

- Otevřít příkazovou řádku ve vhodném adresáři
- Vytvořit lokální repozitář: `git clone xxx`
- Soubor `README.md` přidán do verzování: `git add README.md`
- Uložení změny v `README.md` do lokálního repozitáře: `git commit -m "zpráva"`
- Uložení změny z lokálního do centrálního repozitáře: `git push`
- Stáhnutí změn z centrálního do lokálního repozitáře: `git pull`

Základy gitu - shrnutí



- **git clone cesta**
 - Vytvoří lokální repozitář svázaný s centrálním
 - Provádí se jednou pro daný stroj
- **git pull**
 - Stáhne nejnovější verze souborů z centrálního repozitáře do lokálního
 - Provádí se při přechodu na váš jiný pracovní stroj a nebo pokud někdo jiný modifikuje souboru v centrálním repozitáři
- **git add soubor**
 - Ne všechny soubory má smysl verzovat (např. *.obj, *.class...)
 - Provádí se jednou pro každý soubor, který chcete verzovat
- **git commit -m "description of code change"**
 - Vloží do lokálního repozitáře změny ve všech verzovaných souborech
 - Provádí se často po dokončení logického kusu kódu (několikrát denně)
- **git push**
 - Vloží změny z lokálního repozitáře do centrálního do větve (branch) master nebo main
 - Typicky existují alespoň dvě větve: *master/main* (stabilní kód) a *devel* (aktuální vývoj)
 - Provádí se relativně často (např. každý den)

.gitignore

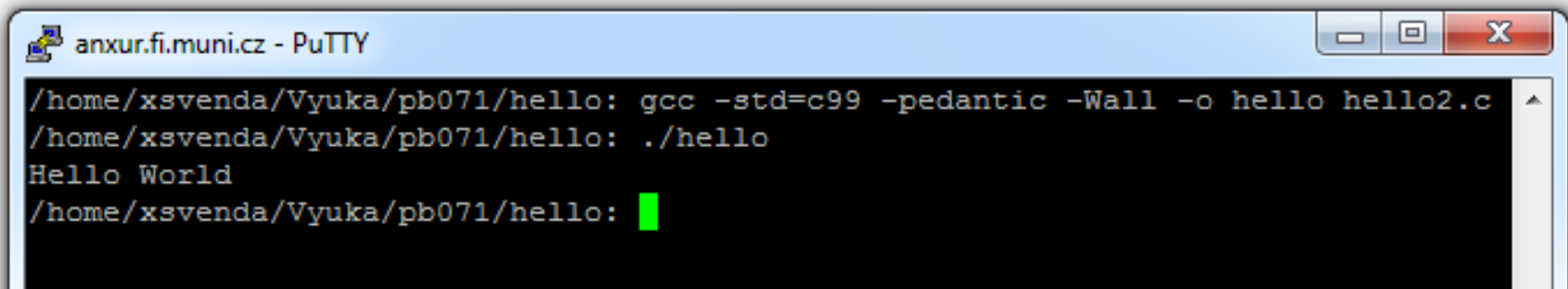
- Speciální soubor gitu ve vašem adresáři
- Specifikuje, které lokální soubory se **nemají** posílat do repositáře (mezivýsledky, videa...)
- U domácích úloh máme předpřipravený *.gitignore*
 - Automatická kontrola přítomnosti před odevzdáním

```
# Intermediate compilation files
*.o
# Resulting binary
*.exe
# Any other file(s)
dont_commit_this.txt
# Ignore whole directory
/pb071_Qt_4_7_4_for_Desktop_-_MinGW_4_4__Qt_SDK__Debug/
/pb071_Qt_*/
```

KOMPILACE NA AISE

Kompilace Aisa

- GNU GCC
 - přepínače (-c, -g, -Wall, -Wextra, -o ...)
 - <https://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 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: █
```

Překlad po částech

1. **Preprocessing** "gcc -E hello.c > hello.i"
 - rozvinutí maker, expanze include...
2. **Kompilace** "gcc -S hello.i" (lépe gcc -std=c99 -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

66

Překlad po částech - preprocessing

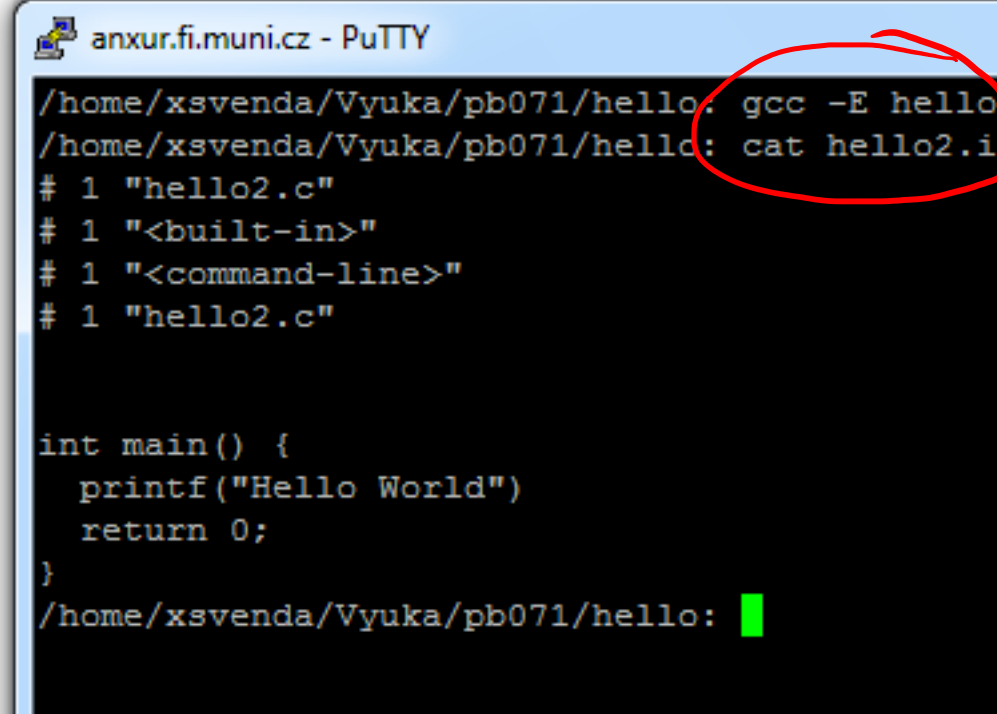
1. Preprocessing "gcc -E hello2.c > hello2.i"

- rozvinutí maker, expanze #include
- odstranění poznámek

hello2.c

```
#define PRINT_MESSAGE "Hello World"

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



```
anxur.fi.muni.cz - PuTTY
/home/xsvenda/Vyuka/pb071/hello: gcc -E hello2.c > hello2.i
/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" (lépe gcc -std=c99 -S hello.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(void) {
    printf("Hello World")
    return 0;
}
```

```
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 declarati
hello2.c:5:3: error: expected ';' before 'return'
/home/xsvenda/Vyuka/pb071/hello: █
```

```
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:
```

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:
```

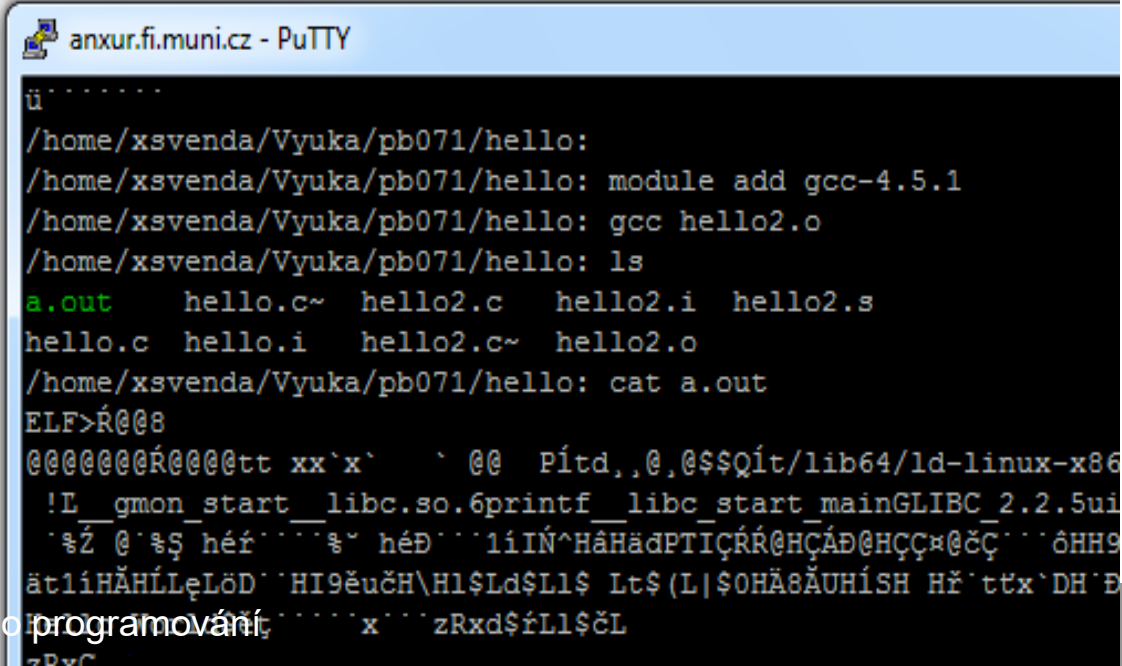
hello2.o

```
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>0@@
UHíz,č,ĚHello WorldzRx
GCC: (GNU) 4.1.2 20080704 (Red Hat 4.1.2-50).syntab.strtab.shstrtab
ta.bss.rodata.rela.eh_frame.comment.note.GNU-stack @0
ε\\1\
>h9Ř
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
!L_gmon_start_libc.so.6printf__libc_start_mainGLIBC 2.2.5ui
`%Z @`%$ hér` ` ` %` héĐ` ` 1iIN^HâHädPTIÇRR@HÇÁĐ@HÇÇ×@čÇ` `ôHH9
ätliHÄHLLeLÖD`HI9ëuĈH\Hl$Ld$Li$ Lt$(L|SOHÄ8ÄUHÍSH Hř`ttx`DH`Đ
zRxC
```

PB071 Prednaska 01 - Kompilace

The Kahoot! logo is centered on a background of a 2x2 grid of colored squares. The top-left square is orange, the top-right is light blue, the bottom-left is yellow, and the bottom-right is green. The word "Kahoot!" is written in a large, white, bold, sans-serif font with a slight drop shadow, spanning across the grid.

- 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 <https://www.fi.muni.cz/pb071/tutorials/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

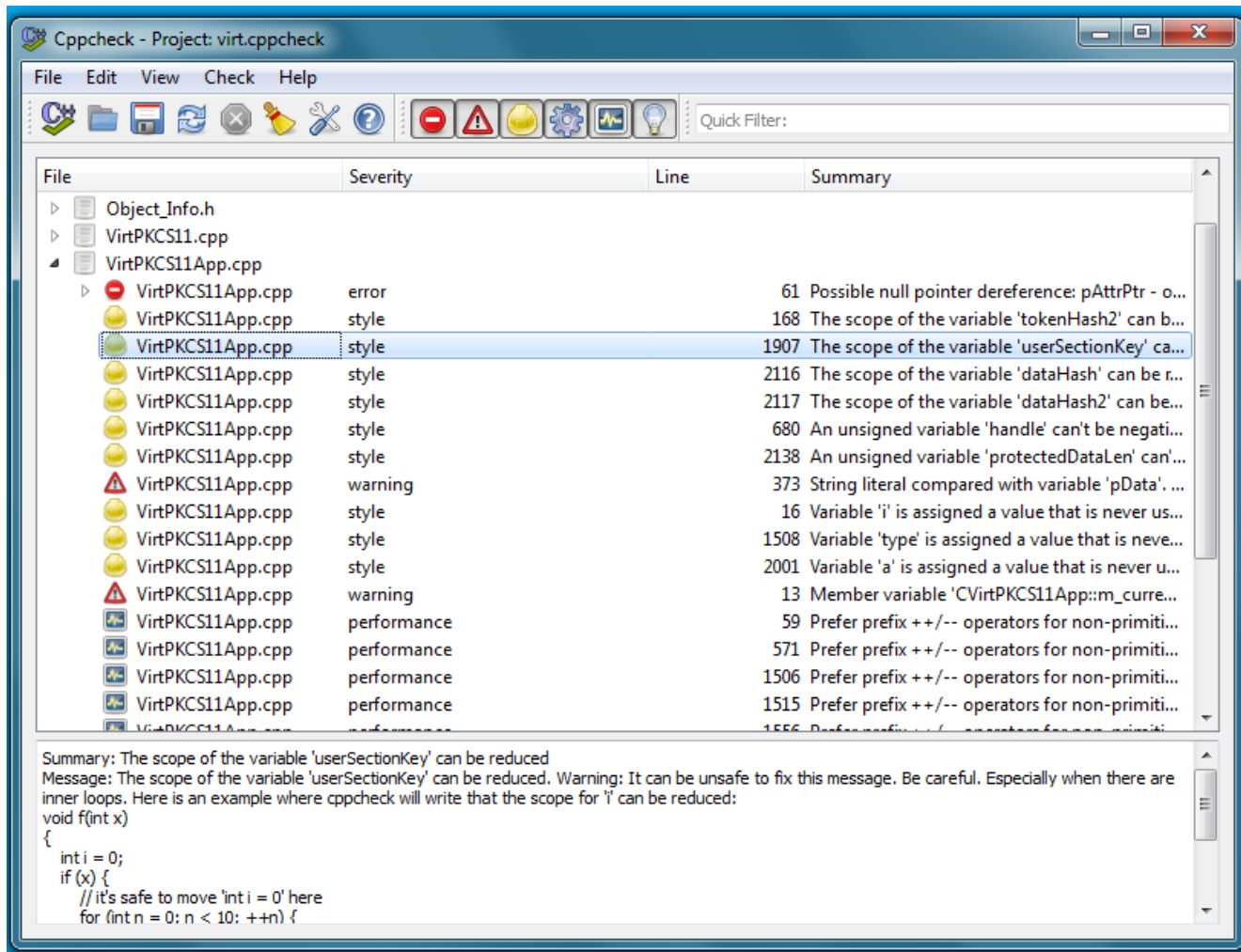
    ...
}
```


Cppcheck



- Nástroj pro statickou analýzu C/C++
 - Open-source freeware, <http://cppcheck.net/>
- Aktivně vyvíjen, poslední verze 2.10 (2023-01-28)
- Široce používaný nástroj
- Příkazová řádka i GUI
- Samostatně stojící verze, plugin do IDE, plugin do verzovacích nástrojů...
 - Code::Blocks, Codelite, Eclipse, Jenkins...
 - Tortoise SVN...
- Multiplatformní (Windows, Linux, iOS)
 - `sudo apt-get install cppcheck`

Cppcheck



Shrnutí

- Organizační – vše na <https://www.fi.muni.cz/pb071>
- Hlavním cílem předmětu je programovat a nadchnout
- Používejte nástroje (IDE, git...)
- Domácí úkoly zadávány/probírány na cvičeních
 - odevzdání na Aise
- Nebojte se zeptat!
 - přednáška, cvičení, poradci, konzultačky...



Anonymous

0 👍

Pokud přeložím program překladačem gcc na Linuxu, bude spustitelný i na Windows?

- Pokud budete mít během poslechu přednášky dotaz, tak jej vložte na [slido.com \(#pb071_2024\)](https://slido.com/#pb071_2024)
- Společně projedeme dotazy na hodině

Join at
slido.com
#pb071_2024

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

F2C – demo (K&R)

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

```
#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 zkompileovat a další problémy

78

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

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;
}
```


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;
}
```

Co si můžete hned doma vyzkoušet

- Připojte se na Aisu a zkuste kompilaci s gcc
- Pohrajte si s git
 - založte si git repozitář na gitlab.fi.muni.cz
- Nalad'te si <https://www.se-radio.net/> 😊