CC Agenda

CSC 100 - Spring 2022

Table of Contents

1 README

This file contains the agenda overview (what I had planned), the objectives (what we managed to do) and (much of the) content of each taught session of the course. I want to avoid splitting the content up over many files - so that you have to navigate as little as possible (like a book)!

The companion file to this file, less structured and with the captain's log, is the notes.org file.

2 Welcome to the course - w1s1 (01/12/22)

2.1 Welcome

- Introduction to the course & the lecturer
- Homework assignment: GitHub Hello World
- What's next?

2.2 Entry survey (Google Forms)

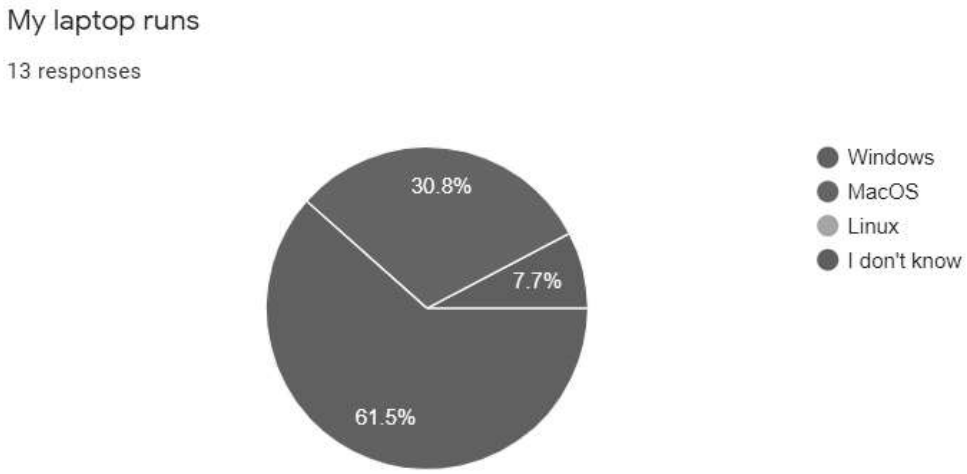


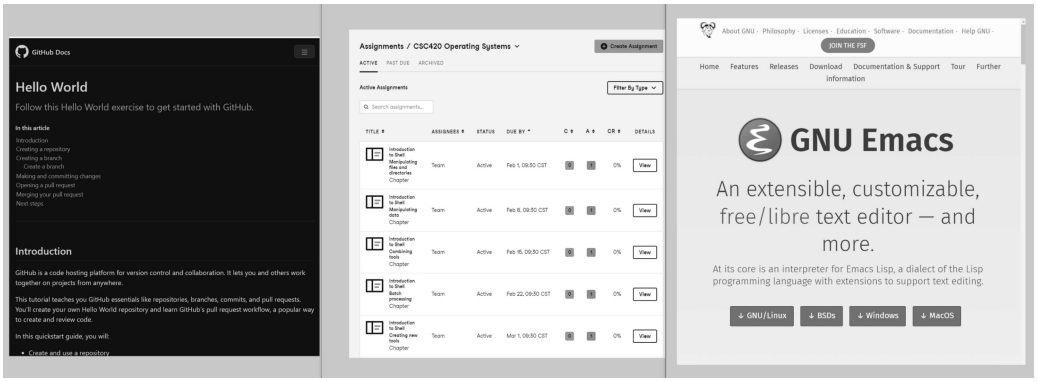
Figure 2: What's your operating system? (Spring 2022 survey)

2.3 Introduction to the course & the lecturer

- PhD theoretical particle physics / WWW development
- C/C++ since ca. 1990 (created multigrid library)

- Professor, Business Informatics @Berlin Univ
- Visiting Assoc Prof for Data Science @Lyon (2021-23)
- Syllabus for this course (Schoology)

2.4 Homework assignments week 1 (14-Jan-2022)



- **GitHub Hello World Exercise (Info: FAQ) - by Friday 14-Jan!**

2.4.1 GitHub

1. What is it?
   - Software development platform (like GitLab, BitBucket, SourceForge, etc.)
   - Built around Git by Linus Torvalds
   - Bought by Microsoft in 2018 (like OpenAI - home of GPT3)
   - 77 mio users (developers) + 200+ mio software projects
   - AI support (e.g. GitHub Copilot - AI-enabled code generator)

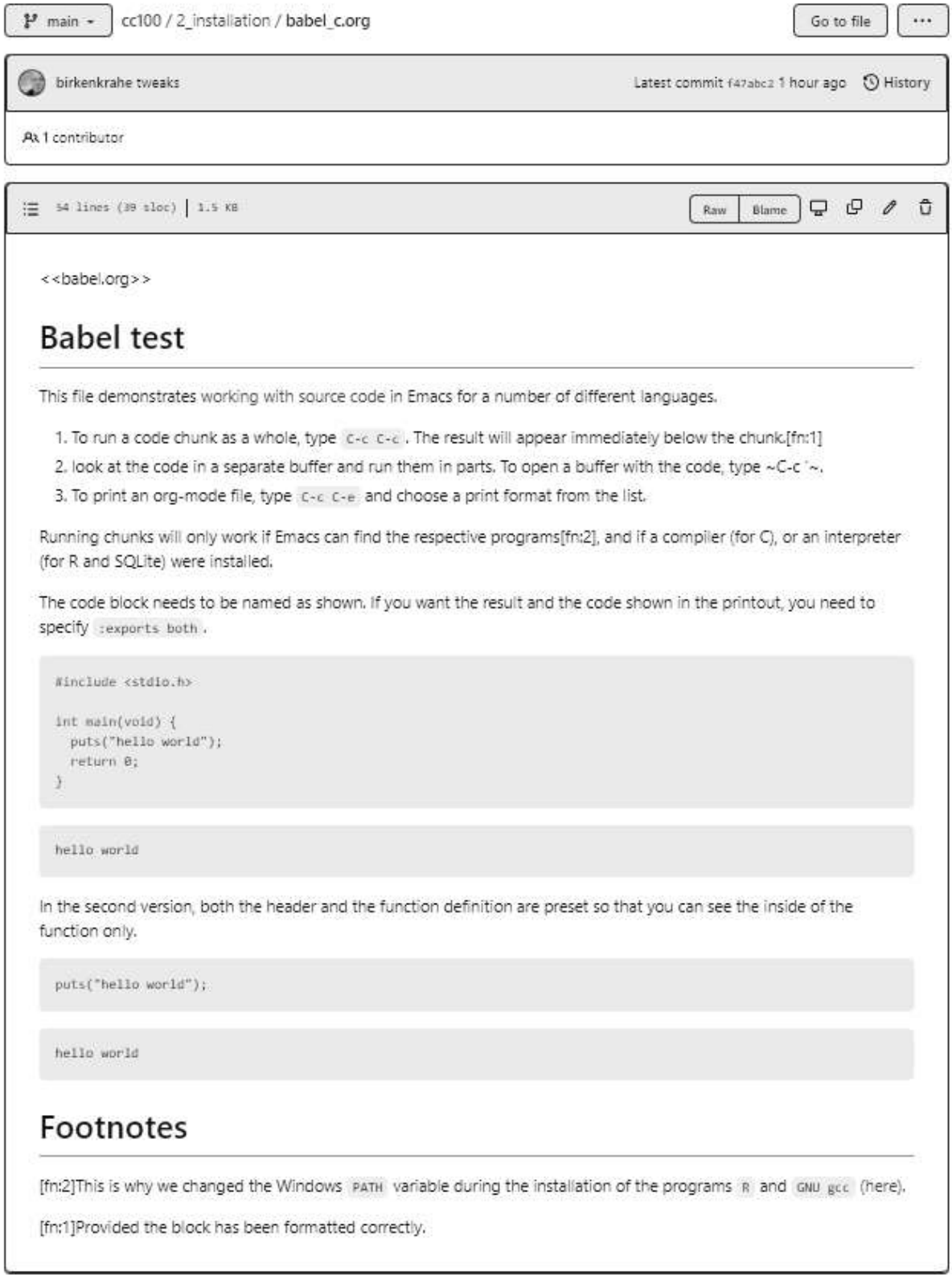
   Watch: "What is GitHub?" (GitHub, 2016)

   Gif: "So long binder of requirements" Source: GitHub

2. Why are we using it?

   Image: Org-mode file in GitHub



main ▾ cc100 / 2_installation / babel_c.org Go to file ···

birkenkrahe tweaks Latest commit f47abc2 1 hour ago History

1 contributor

54 lines (39 sloc) | 1.5 KB Raw Blame

<<babel.org>>

Babel test

This file demonstrates working with source code in Emacs for a number of different languages.

1. To run a code chunk as a whole, type `C-c C-c`. The result will appear immediately below the chunk.[fn:1]
2. look at the code in a separate buffer and run them in parts. To open a buffer with the code, type ~C-c '~.
3. To print an org-mode file, type `C-c C-e` and choose a print format from the list.

Running chunks will only work if Emacs can find the respective programs[fn:2], and if a compiler (for C), or an interpreter (for R and SQLite) were installed.

The code block needs to be named as shown. If you want the result and the code shown in the printout, you need to specify `:exports both`.

```
#include <stdio.h>

int main(void) {
  puts("hello world");
  return 0;
}
```

```
hello world
```

In the second version, both the header and the function definition are preset so that you can see the inside of the function only.

```
puts("hello world");
```

```
hello world
```

Footnotes

[fn:2]This is why we changed the Windows `PATH` variable during the installation of the programs `R` and `GNU gcc` (here).

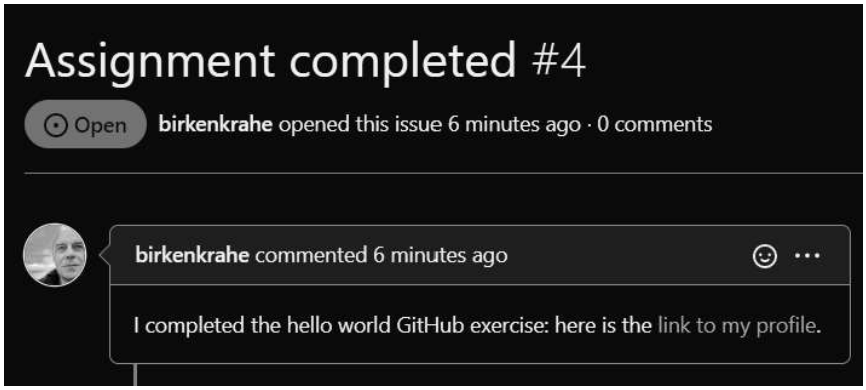
[fn:1]Provided the block has been formatted correctly.

- It's free
- To host course materials
- Upload assignments (esp. Emacs Org-files)
- Discussion
- Wiki for collaboration
- Complements Schoology

3. What will you have to do?

- Sign up with GitHub - use Lyon Email
- Pick an available username **using your own first and last name**
- Complete the "Hello World" exercise (FAQ)
- Create an issue from the `cc100` repository like in the example below (except from your account instead of mine).

Image: Issue "Assignment completed"



If you do have a GitHub account already, do the exercise anyway using your existing account (it takes 10 min)! Make sure you let me know what your user name is so that I can add you to my repo.

4. What else can you do?
   - You can fork the `cc100` repository
   - You can watch the `cc100` repository - and set `Notifications` to `Participating and @mentions` so that you see my comments (see image below).

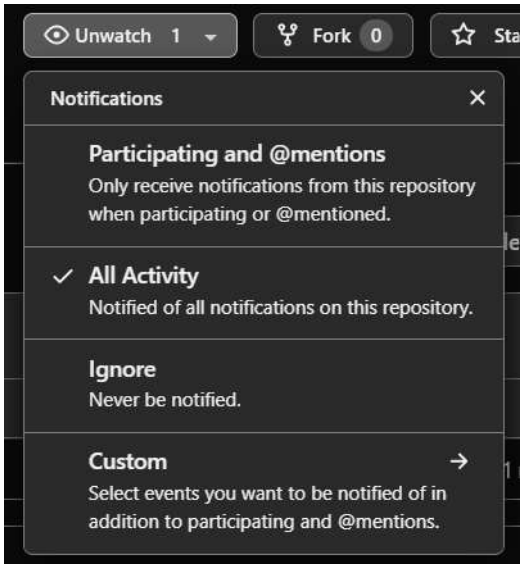
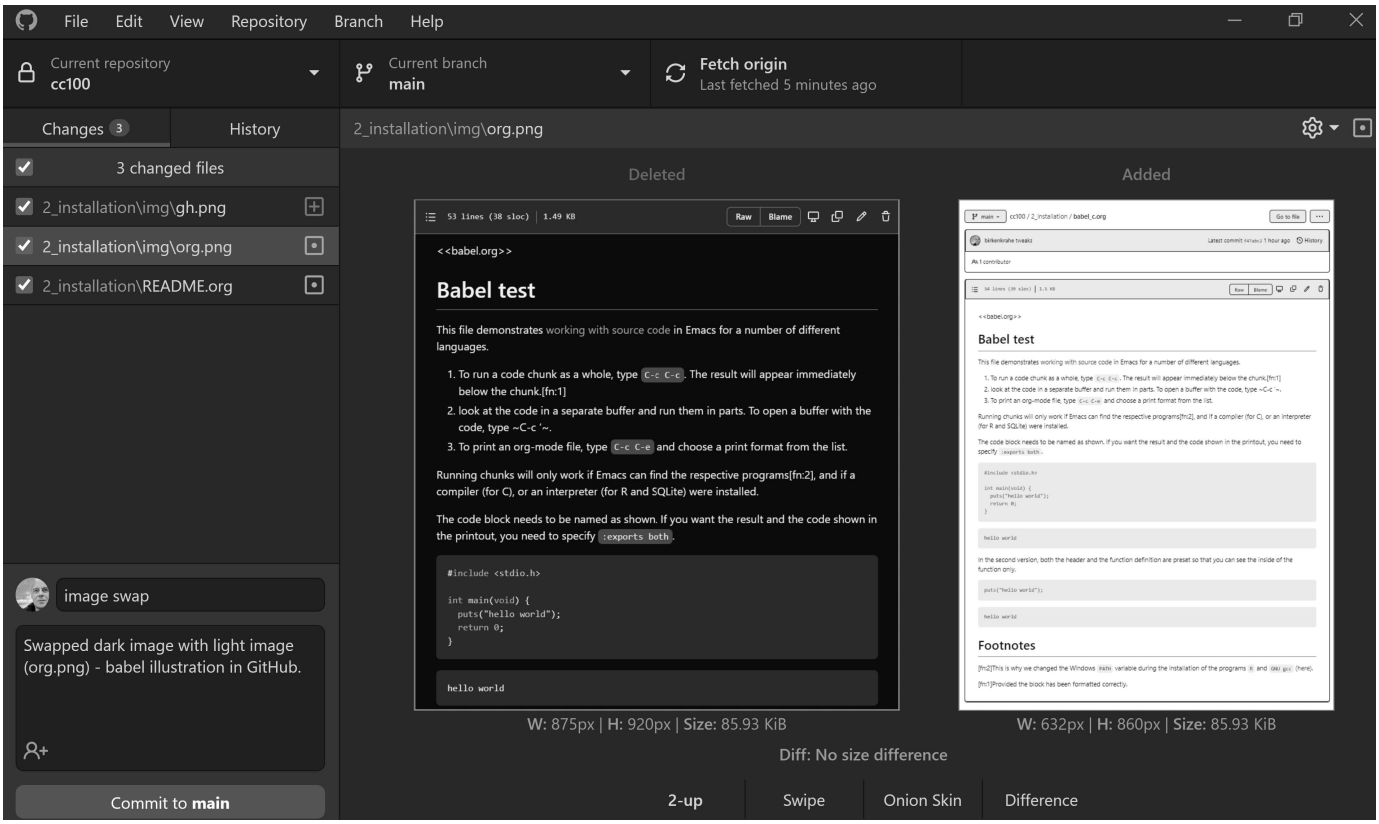


Image: Notifications settings when watching a repository

   - You can submit issues from the repository (e.g. if you notice mistakes or if you want extra information, or to share a link)
   - You can participate in discussions (sometimes I will make you)
   - You can add to the wiki (e.g. comments and links to interesting resources)
   - You can use it as a platform for projects or coding
   - You can download the desktop client to manage repos on your PC (see image below).

Image: GitHub desktop client commit



2.5 What's next?

- See schedule (GitHub)
- Watch online lecture on "Systems" (to be done)
- Later: online summary (`notes.org` in GitHub)
- Sometimes: diary notes (`diary.org` in GitHub)
- Class on Friday 14-Jan will be online!
- Hope to see you at school next Monday!

3 Git, GitHub, History of C - w1s2 (01/14/22)

3.1 Overview

HOW	WHAT
Review	GitHub Hello World exercise (see FAQ)
Lecture	Introduction to C
Practice	Install C compiler (see FAQ)
	Set PATH environment variable
	Test C compiler

3.2 Objectives

- [X] Review the basics of Git and GitHub
- [X] Understand what C is, and why you learn it
- [ ] Install the GNU C and C++ compiler (`gcc`)
- [ ] Set `PATH` environment variable under Windows
- [ ] Test the C compiler

4 Introduction to C and C++ - w2s3 (01/19/22)

4.1 I'm back

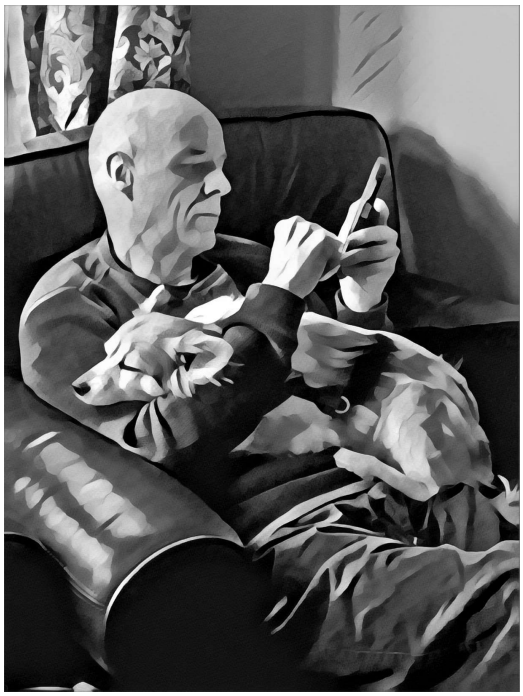


Figure 9: "I'm back, baby."

4.2 Overview

HOW	WHAT	TIME

HOW	WHAT	TIME
Review	Quiz on last week's content	15'
	Quiz: feedback discussion	5'
Lecture	Introduction to C (cont'd)(gh)	10'
	Installation of tools (gh)	
Practice	Install C compiler[1] (see FAQ)	
	Set PATH environment variable	
	Test C compiler	

gh = GitHub link

4.3 Objectives

- [X] Review last week & discuss & get feedback on quiz 1
- [X] Complete introduction to C (and C++)
- [ ] Understand installation process (philosophy)
- [ ] Install the GNU C and C++ compiler (`gcc`)
- [ ] Set `PATH` environment variable under Windows
- [ ] Test the C compiler

5 Install C compiler, set PATH - w2s4 (01/21/22)

5.1 Overview

HOW	WHAT	TIME
Lecture	Installation of tools (gh)	
Practice	Install C compiler[2] (see FAQ)	
	Test C compiler `gcc --version`	
	GNU Emacs tutorial in class (gh)	
HOME	Set PATH environment variable	

gh = GitHub link

5.2 Objectives

- [X] Understand installation process (philosophy)
- [X] Install the GNU C and C++ compiler (`gcc`)
- [X] Set `PATH` environment variable under Windows
- [X] Test the C compiler
- [X] GNU Emacs tutorial

6 Install GNU Emacs - w3s5 (01/24/22)

6.1 Overview

HOW	WHAT
Practice	Emacs training script
	See also video playlist

HOW	WHAT
Assignment	Set `.emacs` init file in your home directory

6.2 Objectives

- [X] Work through short tutorial for GNU Emacs
- [X] Explain Emacs assignment

7 Create GNU Emacs Org-mode file - w3s6 (01/26/22)

7.1 Overview

HOW	WHAT
Practice	Set `.emacs` init file in your home directory
Demo	Creating Emacs Org-mode with C code and run it
Assignment	Create Emacs Org-mode file (GitHub)

7.2 Objectives

- [X] Understand Emacs initialization with `.emacs`
- [X] Learn how to create an Org-mode file
- [ ] Run a C program inside Emacs

8 GNU Emacs initialization file - w3s7 (01/28/22)

1. We continue where we left it on Wednesday (w3s6)
2. Fixing the .emacs problem (FAQ)
3. Finish Org-mode assignment (GitHub)
4. Submit results to GitHub as issue (ZIP) - by 11.59PM tonight

9 GCC Lab session - w4s8 (01/31/22)

Figure 10: Teaching Emacs on Dagobah

We will hold a special lab session tomorrow, Monday 31 January 11-11.50 AM, to sort out any issues related to Emacs or GCC. Bring your own PC to the session, or work on a lab desktop. I will spend the time going round to make sure that you can

- Install/ open / use the Emacs editor

- Create, run and tangle Org-mode files
- Install / use the C compiler GCC
- Understand the recent program assignments

The necessary steps are also demonstrated in this tutorial video playlist.

We will continue with our regular program on Wednesday, 2nd February at 11 AM - a short quiz will be available before.

For those who know or can do all of this already: here's a second challenge (with solution) to practice while I sort others out.

10 Structure of a C program - w4s9 (02/02/22)

10.1 Overview

HOW	WHAT	WHEN
Lecture	C Fundamentals (King ch. 2)	
Practice	Write and execute `pun.c`	
Assignment	Write a checkmarks program	Friday, 4 Feb, 11AM

10.2 Objectives

- [X] Understand the basic structure of a C program
- [X] Write a simple, complete C program (`pun.c`)
- [X] Submit simple assignment for Friday 4 Feb 11 AM

10.3 Assignment: checkmark program

Submit program and output as an Org-mode file. It should look like this (code block is folded):

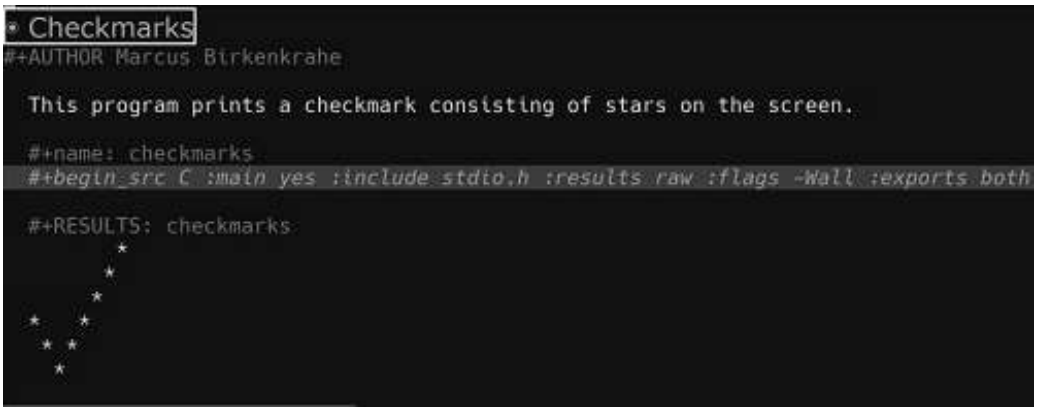


Figure 11: Checkmarks solution (code block folded)

11 Variable type declarations and assignments - w5s10 (02/07/22)

11.1 Overview

HOW	WHAT	WHEN
Review	Structure of a program	

HOW	WHAT	WHEN
Lecture	C Fundamentals (King ch. 2)	
Practice	Computing the weight of a box	
Test 1	10 from Quiz 1-3 + 10 new MPCs	Friday, 11 Feb, 11AM

11.2 Objectives

- [x] Understand the framework for the first test
- [x] Understand variable, data types and type declarations
- [x] Understand variable assignments
- [ ] Understand printing formats
- [ ] Write a program with variable declarations and assignments
- [ ] Understand printing formats

12 Formatting printout - w5s11 (02/09/22)

12.1 Overview

HOW	WHAT	WHEN
Lecture	C Fundamentals (King ch. 2)	
Practice	Interactive notebook: `printf` formatting	
Test 1	10 from Quiz 1-3 + 10 new MPCs	Friday, 11 Feb, 11AM

12.2 Objectives

- [x] Understand printing formats
- [x] Write a program with variable declarations and assignments

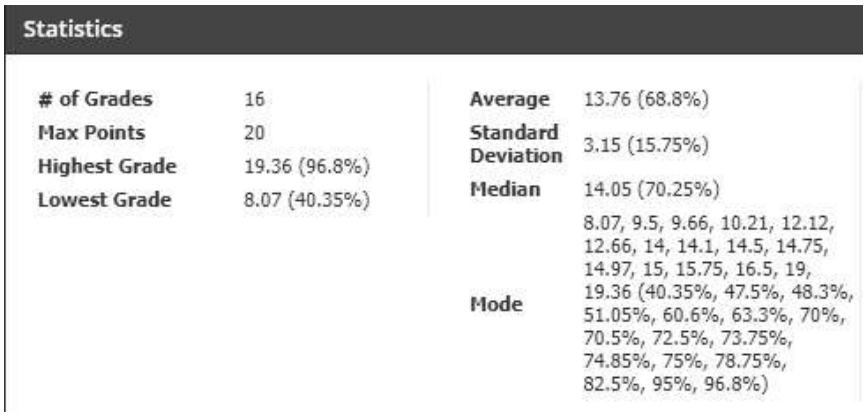
13 Test review - w6s12 (02/14/22)

13.1 Objectives

- [x] Understand test results
- [x] Know what to do different next time
- [x] Discuss selected questions and answers

13.2 Test results - stats and plots

- The results are nothing to write home about - though > 50% means that the class passed (on average).



Statistics

# of Grades	16	Average	13.76 (68.8%)
Max Points	20	Standard Deviation	3.15 (15.75%)
Highest Grade	19.36 (96.8%)	Median	14.05 (70.25%)
Lowest Grade	8.07 (40.35%)	Mode	8.07, 9.5, 9.66, 10.21, 12.12, 12.66, 14, 14.1, 14.5, 14.75, 14.97, 15, 15.75, 16.5, 19, 19.36 (40.35%, 47.5%, 48.3%, 51.05%, 60.6%, 63.3%, 70%, 70.5%, 72.5%, 73.75%, 74.85%, 75%, 78.75%, 82.5%, 95%, 96.8%)

Figure 12: Test 1 results (Schoology)

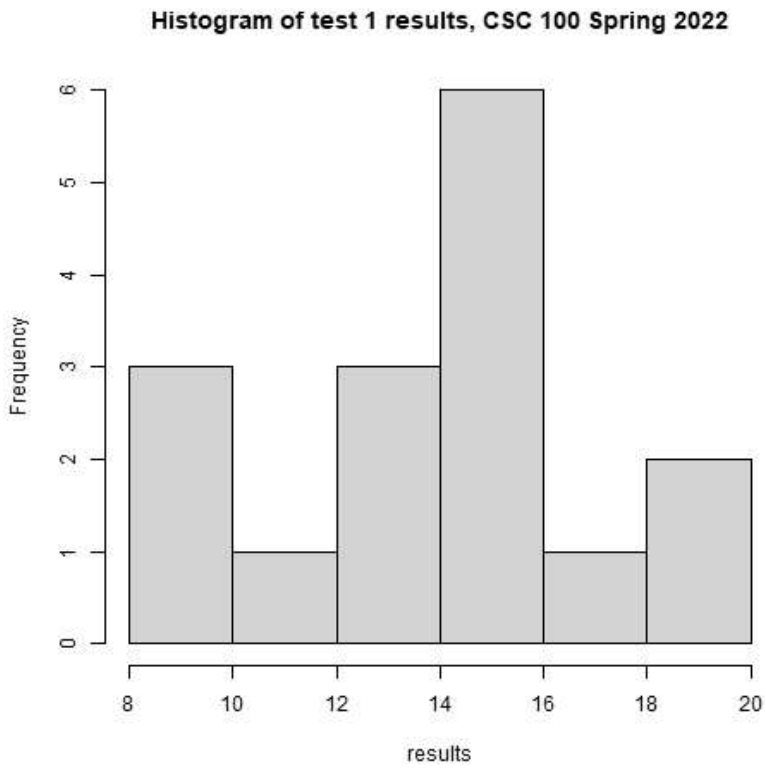
- I am an obsessive fact-checker. When checking the stats with R, I find slightly different results:

```
results <- c(8.07,9.5,9.66,10.21,12.12,12.66,14,14.1,
             14.75,14.75,14.97,15,15.75,17,19,19.36)
sd(results)
summary(results)
```

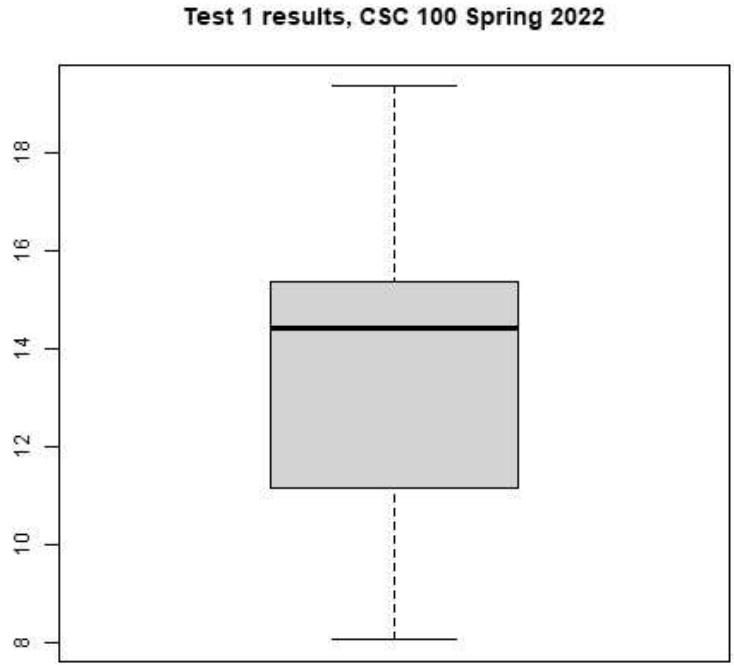
```
[1] 3.288485
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   8.07   11.64   14.43   13.81   15.19   19.36
```

- Let's make some plots: histogram, boxplot and density plot. I'dd like the histogram and the density plot (a smoothened histogram) to peak more to the right, and for the boxplot to be smaller and higher up.

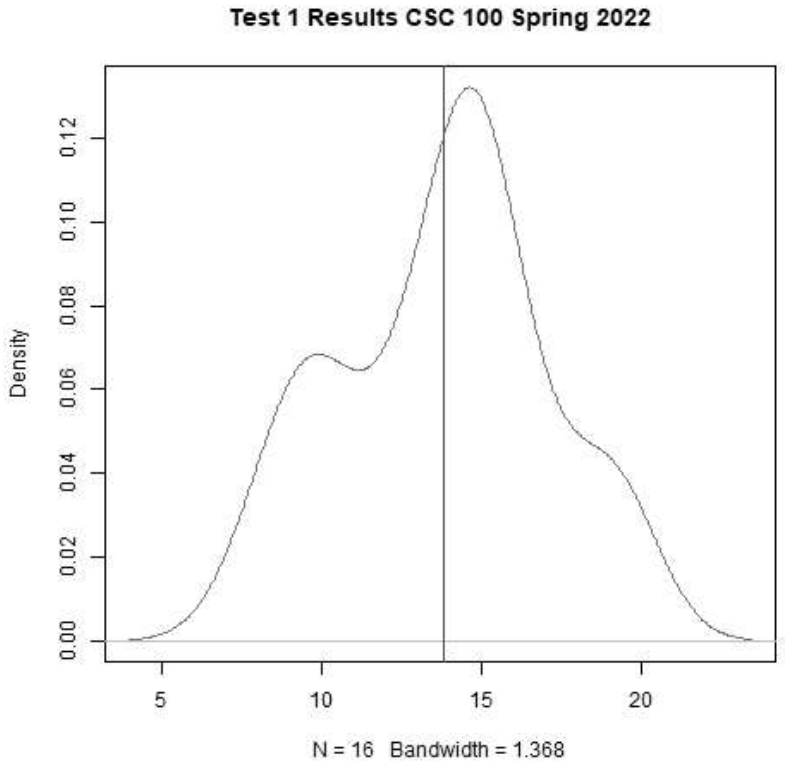
```
hist(results, main="Histogram of test 1 results, CSC 100 Spring 2022")
```




```
boxplot(results, main="Test 1 results, CSC 100 Spring 2022")
```



```
ave <- mean(results)
d <- density(results)
plot(d, col="steelblue",main="Test 1 Results CSC 100 Spring 2022")
abline(v=ave,col="red")
```



13.3 Analysis - feedback and action points

- Test 1 can now be played an unlimited number of times. I have added feedback to all new questions.
- There will not be another paper-based test: the results weren't much better than in other courses, and test preparation and grading are excruciating if partial credit is given.
- What surprised me most was that many of you did not use the available time.
- See also: "I can teach it to you but I cannot learn it for you"
- Questions:
  - How did you study for this test?
  - If you didn't perform well, what will you change?
  - What can I do to help you help yourself?
- CHANGES TO BE APPLIED BY BIRKENKRAHE (FUTURE QUIZ/TESTS):
  - Fewer choices for the multiple choice
  - Announce if a question has > 1 answer (and how many)

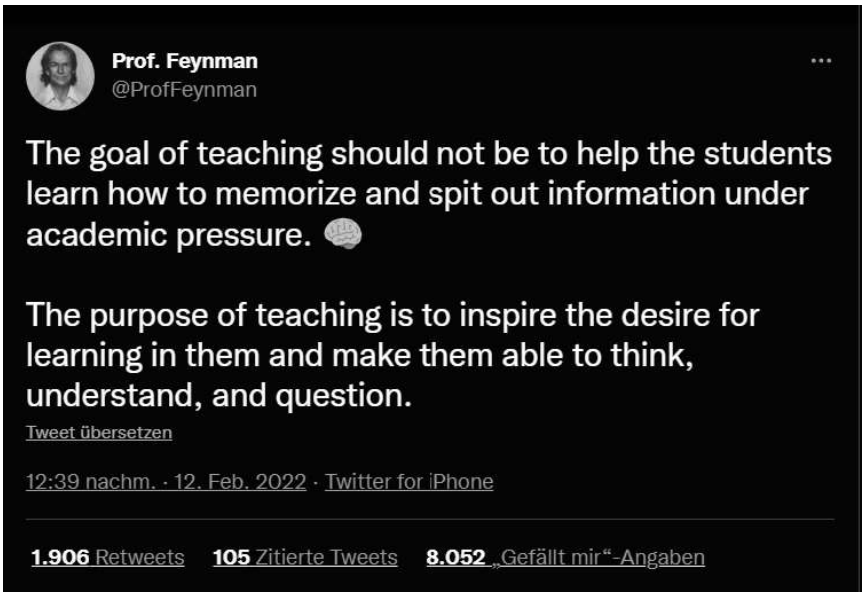


Figure 16: Feynman (via Twitter)

- Post-mortem on a couple of questions
  - comment format (last question)
  - `printf` format

```
/*     one line
 *     multi line
 */
float f = 4.0f;
printf("hello %.pf", f);
/* one line
   or multi line
*/
```

13.4 Test questions and answers

- We go through all questions together
- Everybody can contribute an answer
- Write down questions and ask them now!

13.5 This week (6)

- Test review
- Program assignment
- Defining constants
- Naming identifiers
- Reading input

13.6 Program assignment (until Monday 21-Feb, 11AM)

Compute the volume of a sphere

- Write a program that computes the volume of a sphere with a 10-meter radius, using the formula $v = 4/3 \times \pi r^3$.
- Write the fraction $4/3$ as `4.0f/3.0f`. (Try writing it as 4/3 and see what happens.)
- Remember that C does not have an exponentiation operator, so you need to write r^3 as `r*r*r`.
- Upload your solution program as a `.c` file or as a `.org` file to Schoology not later than 11 AM on Monday, February 21st. Make sure that your program actually runs without errors!

- Be prepared to present your solution in class.

14 Formatting, C constants - w6s13 (02/16/22)

14.1 Objectives

- [X] Review: be the compiler / function challenge
- [X] Understand different ways of defining C constants
- [ ] Practice constant definitions
- [ ] Understand naming conventions for C programs

14.2 Warming up: be the compiler!

The output of the program below should be:

```
: The tax on 200.00 is 35.00, so the grandtotal is 235.00.
```

but instead the output is

```
: The tax on -1093221452 is 0, so the grandtotal is 1080623104.
```

Can you fix the program 1?.

```
float subtotal;
int tax;
float grandtotal;
float taxrate;

taxrate = 0.175;
subtotal = 200;
tax = subtotal * taxrate;
grandtotal = subtotal + tax;

printf("The tax on %d is %d, so the grandtotal is %d\n",
       subtotal, tax, grandtotal);
```

14.3 Programming challenge: hello world function

- Things to remember:
  - You need to declare stuff before you can use it in C.
  - The function definition looks just like `main`
- Challenge: Write a function `hello()` that prints out `hello world` when called.
- You can do this in Emacs in one code block, which contains the definition and the function call.
- This challenge carries extra credit! Submit Org-file via email.
- Solution next week! (This implies that you send it to me before the first class next week.)

15 C constants, naming - w6s15 (02/18/22)

15.1 News

- Invitation for an online data science seminar with the data scientist of Stone Ward, Little Rock/Chicago, Matthew Ward: join us at 3 pm via Google Meet (see Schoology) - incl INTERNSHIPS

- Great tips for getting better with Emacs quickly in this post. See also my Emacs for Beginners YouTube playlist.

  > I started as a rank newbie 14 days ago, and I've learned enough to "get started", kind of grok emacs beginner fundamentals and simple navigation, and I feel VERY good about my progress. I wanted to share how I got to this point. (Note that I happened to have the time to read/watch/practice a few hours a day. If I only had 30 min per day, it would have taken me a month or two to get to this level.)

15.2 Objectives

- [X] Learn more ways of defining C constants
- [X] Practice constant definitions
- [X] Understand naming conventions for C program

15.3 Warming up: be the compiler!

- What's wrong with this program? (It won't even compile.)

```
#include <stdio.h>
#define CONST_ 100;

int main(void) {
  int value = CONST_, newValue;

  newValue = value + 1;

  printf("%d %d %d\n", CONST_, value, newValue);
  return 0;
}
```

  - Answer in the class notes file `notes.org`.

15.4 Next week

- Quiz 4 (play me once, then play me often!)
- Review of the volume/hello programming challenges
- A new programming challenge!
- Reading input with `scanf` (read me, baby!)
- Good program layout
- Arithmetic operators (math, yay!)

16 Program assignment review, scanf - w7s16 (02/21/22)

16.1 News

- Script is always in GitHub (use it for your class post mortem)
- Literate Programming: Empower Your Writing with Emacs Org-Mode

16.2 Objectives

- [X] Review programming assignment "volume"
- [X] Review programming challenge "hello world function"
- [X] Understand reading input (from the command line)
- [ ] Understand how in-class assignments work
- [ ] New programming exercise "phone number conversion"
- [ ] Complete two in-class assignments
- [ ] See how reading input works in Emacs Org-mode

16.3 Warming up: be the compiler!

- The following program should print out `3.0` but it does not even compile! Find all mistakes, fix them and run the program in Emacs!

```
float int1=1.f, int2=2.f, int;
printf("%.2f\n"; int = int 1 + int 2);
```

- SOLUTION: see class notes.

16.4 How do class assignments work?

- In-class assignments are **10%** of your total grade
- They are labeled **class assignments** in the Schoology gradebook
- You get the points if you attend and participate **actively**
- If you check your phone instead, you're **not** active
- If you could not attend, submit **late**

16.5 Programming assignment review

- I left detailed comments for most of you: use them or ask me! Use them by modifying your submission accordingly.
- You can find examples and solutions in the class notes later.
- Future submissions will be graded not just for effort but for accuracy (this includes following the instructions). However, you can (and should) always re-submit an improved version.
- Computer science is a craft like any other: (1) you keep at it until you got it right, (2) you don't give up easily, and (3) you ask for help when you're at the end of your wits.
- Please submit an Org-mode file if you have one so that I don't have to check your results line-by-line but can just run it.

- Org-mode meta data were often not correct (file will still run)

```
#+AUTHOR: name

#+name: code
#+begin_src C
...
#+end_src
```

- A couple of programs looked awfully similar (down to errors):
  - There is no need to bend the rules: this is not a rat race!
  - If in doubt about "what is cheating", check the syllabus.
  - Add `#+HONOR: pledged` to the top of your Org-mode submissions
- You can find online REPLs (Read-Eval-Print-Loops) like this one to try things out if you're flummoxed by Emacs Org-mode (at this point, you shouldn't be but Windows can be toxic…)

16.6 Challenge review

- User-defined functions are where programming really begins
- See class notes for challenge solution

17 Reading input with Emacs, tangling code - w7s17 (02/23/22)

17.1 Objectives

- [X] New programming exercise "phone number conversion"
- [X] How to use `scanf` inside Emacs (and other input)
- [X] Understand reading input (from the command line)

17.2 How to use `scanf` inside Emacs with redirection

- Emacs can fetch arguments via the header arguments `:cmdline` (as input redirected from a file), or via `:var` through assignment.

```
float i,j;
scanf("%f %f\n",&i, &j);
printf("input was: %f %f\n",i,j);
```

- [X] If your neighbor does not get along, help him/her without being asked explicitly! (Well done everyone!)
- [X]

  Create a file called `input` in the current working directory. It contains only the numbers `1.` and `2.`:

```
1. 2.
```

- [X] Create and run the code block 1 in Emacs in a Org-mode file. Make sure that you have `:main yes` and `:include <stdio.h>` either on the code block header line or in the `#+PROPERTY:` meta data of your Emacs Org-mode file.
- [X] Create and run the code block 1 in Emacs. Make sure that you have `:main yes` and `:include <stdio.h>` either on the code block header line or in the `#+PROPERTY:` meta data of your Emacs Org-mode file.
- [X] Tangle the code block and look at the file `cmd.c`. To tangle, use the key sequence `C-c C-v t`.
- [X] Open a CMD terminal (or `M-x shell` inside Emacs), and compile the program `cmd.c`. Call the executable `cmd`.
- [X]

  Run the program on the command line like this:

```
cmd < input
```

- Shell "redirection" (directing the content of `input` into `cmd`) is an important shell process. It uses both `stdin` and `stdout`. See GNU bash manual for details.

17.3 Programming assignment (until ~~Monday, 28-Feb~~ Tuesday, 1-Mar, 11 AM)

- Write a program that prompts the user to enter a telephone number in the form `(xxx) xxx-xxxx`, and then displays the number in the form `xxx.xxx.xxxx`.
- Example input/output of the first program, `phone1.c`:

```
Enter phone number [(xxx) xxx-xxxx]: (870) 456-7890
You entered 870.456.7890
```

- Write another program that asks for the input format in the form `xxx\xxx\xxxx`, and then displays the number in the form `(xxx)xxx-xxx`.
- Example input/output of the second program, `phone2.c`:

```
Enter phone number [xxx\xxx\xxxx]: 870\456\7890
You entered (870) 456-7890
```

- Submit one Emacs Org-mode file `phone.org` with both programs in it as code blocks that can be **tangled** as `phone1.c` and `phone2.c`, resp.
- The header information of your Org-mode file should look like this:

```
#+TITLE: Phone number conversion
#+AUTHOR: [your name]
#+HONOR: pledged
```

- Tip: some characters, like `\` are protected because they are part of the file `PATH`. If you want to use them, you have to "escape" them with an extra `\`, like the newline character `\n`. So to print or to scan the character `\`, you use `\\`.

```
printf("hi there\n");              // string output
printf(" \"hi there\" \n");        // escaped \" will print
printf("This is a slash: \ \n");   // slash will not print
printf("This is a slash: \\ \n");  // escaped slash\\  will print
```

18 Lab session: scanf.org, reading input - w8s18 (02/28/22)

18.1 Objectives

- [X] Review: quiz questions 4 + 5 (complete them/ask)
- [X] Be the compiler: `scanf` formatting

- [ ] Review: compile and run C on the command line
- [ ] Understand tangling code once and for all

18.2 Be the compiler: `scanf` formatting

- Download `reading_input.zip` from GitHub (cc100/practice)
- The ZIP file contains an interactive notebook `scanf.org` and several prepared input files
- Work through the notebook at your own pace
- If you cannot finish in class, finish it at home!
- Help thy neighbour and/or ask me for help!

18.3 Compile and run C in the shell

- [ ]

  Create a minimum C file, e.g. a hello world program `hw.c`- either in Emacs or in any other editor you like:

  ```
  #include <stdio.h>

  int main() {
    puts("hello world");
  }
  ```

- [ ] Open a terminal (or `M-x shell` or `M-x eshell` in Emacs).
- [ ] Convince yourself that the file can be found with `DIR hw.c`
- [ ] Compile the file with `gcc -o hello hw.c`
- [ ] The `-o hello` means "name the executable `hello`"
- [ ] Run the file by entering the name of the executable
- [ ] As you can see via `dir hello*`, the executable is actually called `hello.exe` in Windows.

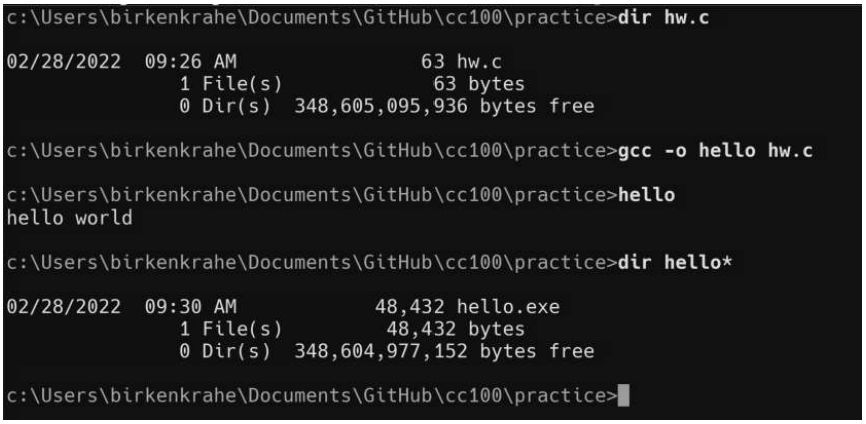


Figure 17: Compile and run C program on the shell

18.4 2 Tips for the programming assignment

1. (02/25/2022) If you want to test and run your file **inside** Emacs, which saves a lot of time, just add this to the header arguments (after the `C`):

   ```
   :cmdline < input
   ```

   where `input` is a file that contains the phone number as requested for input, `(xxx) xxx-xxxx` or `xxx\xxx\xxxx`. Now `C-c C-c` will run and execute the program. Alternatively, you have to tangle the code block, compile and run it by hand on the command line.

2. (02/27/2022) the formatting in `scanf()` needs to match the input pattern. E.g. if the input is given as

```
foo==++//bar
```

(with `foo` and `bar` decleared as `int` variables) then only the following command will pick the input up properly:

```
scanf("%d==++//%d", &foo, &bar);
```

irrespective of the output, which can be formatted in any way we like. `scanf()` only looks for two integers separated by all that junk between the numbers.

Example: the file `inputfile` contains only: `444==++//555`. We want to only print out `444` and `555`.

```
int a,b;
scanf("%d==++//%d", &a, &b);
printf("%d %d", a,  b);
```

19 Compiling, LitProg, Program Layout - w8s19 (03/04/22)

19.1 News

- `[X]` Mid-term grade speech / Improve grade with a project (FAQ)
- `[X]` Program assignment **for extra credit**: PDF (GitHub)
- `[X]` Practice material/notebooks now in GDrive (`reading_input`)
- `[ ]` PDFs of the quizzes with solutions now in GitHub

19.2 Improve your grade with a project

If you want to improve your grade, you can talk to me about doing a small, independent research project leading to a writeup in the form of a notebook, or a short (10-15 min) presentation. The topic must be related to the topic of this course.

19.3 Objectives

- `[X]` Understand how to compile and run C in the shell
- `[X]` Understand literate programming once and for all
- `[X]` Understand what good program layout means (practice)

19.4 Compile and run C in the shell

- Three shells[3] are at your disposal: Microsoft shell (MacOS terminal), Emacs eshell, Emacs shell (= terminal) = CMD line:
  - Start Windows CMDline from the the Windows Search (`CMD`)
  - Start Emacs shell (Windows CMDline in Emacs): `M-x shell`
  - Start Emacs eshell with `M-x eshell`
- Files you run in the terminal: Windows `*.exe` files - executable in binary format = machine code
- `[X]`

  Create a minimum C file, e.g. a hello world program `hw.c`- either in Emacs or in any other editor you like:

```
#include <stdio.h>

int main() {
  puts("hello world");
}
```

```
hello world
```

  You should see the output:

```
#+RESULTS: helloworld
: hello world
```

- [X] Tangle the code block above with `C-c C-v t` (if you have more than one code block in an Org file, use `C-u C-c C-v t`)
- [X] Open a terminal (or `M-x shell` or `M-x eshell` in Emacs).
- [X] Convince yourself that the file can be found with `DIR hw.c`
- [X] Compile the file with `gcc -o hello hw.c`
- [X] The `-o hello` means "name the executable `hello`"
- [X] Run the file by entering the name of the executable
- [X] As you can see via `dir hello*`, the executable is actually called `hello.exe` in Windows.

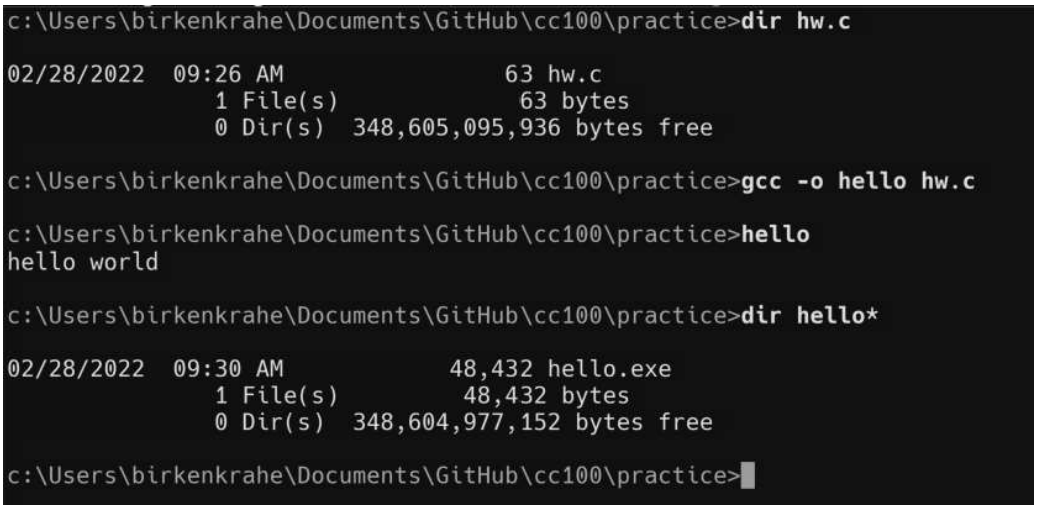


Figure 18: Compile and run C program on the shell

- [X] The general structure of the compile command `gcc` is

```
gcc -o [outfile] [infile]
```

19.5 Understand tangling code once and for all

19.5.1 Concepts

Concept review: make sure that you can answer these questions:[4]

What is Emacs?
What is Org-mode?
What does "tangle" C code mean?
What does "weaving" documentation mean?

19.5.2 Guided tour

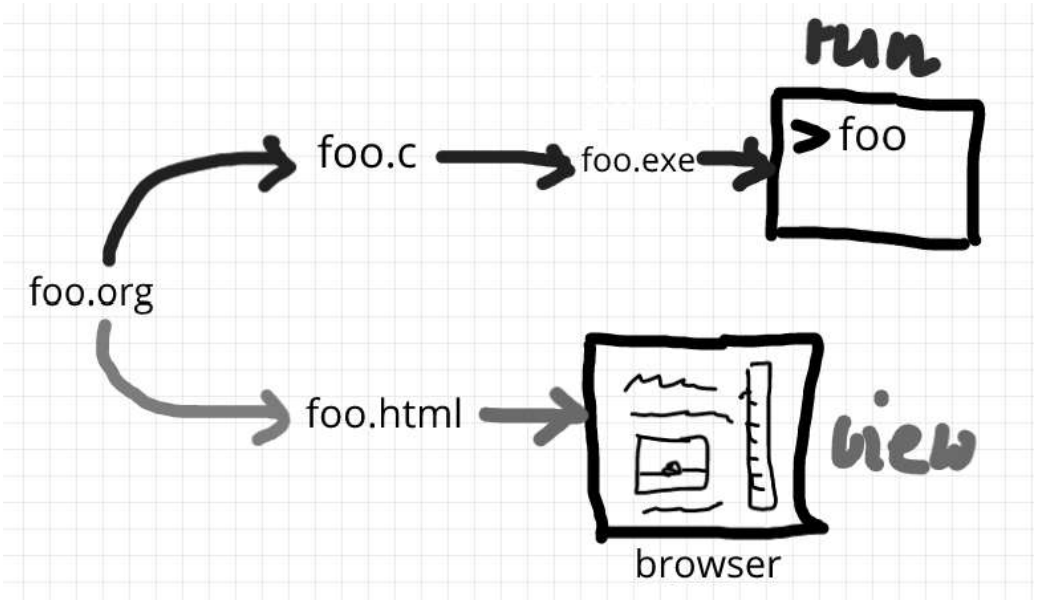
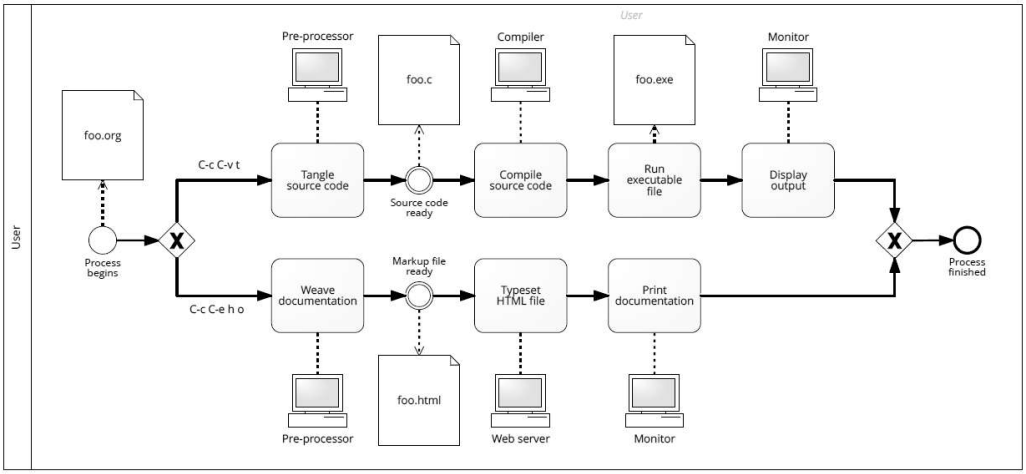


Figure 19: Short explanation of literate programming



Tangle foo.org in Emacs: C-c C-v t Weave HTML page in browser: C-c C-e h o
Compile foo.c in shell: gcc foo.c Run executable foo.exe: $ foo

Figure 20: Detailed description of literate programming for C

```
* The following statement reads an ~int~ value and stores it in the
  variable ~i~. To run it, you have to tangle the code block (with
  ~C-u C-c C-v t~), compile and run it on the command line - not on
  the simulated ~eshell~ or the Emacs ~shell~, but on the Windows
  CMD terminal command line[fn:14].
  #+name: iscan
  #+begin_src C :tangle iscan.c :includes <stdio.h> :main yes :comments both
    int i;
    puts("Enter an integer!");
    scanf("%d", &i);
    printf("You entered %d\n", i);
  #+end_src
1 U\---  README.org     14% (674,17)  Git:main  (Org)
```

Figure 21: Code block and comment in Org-mode

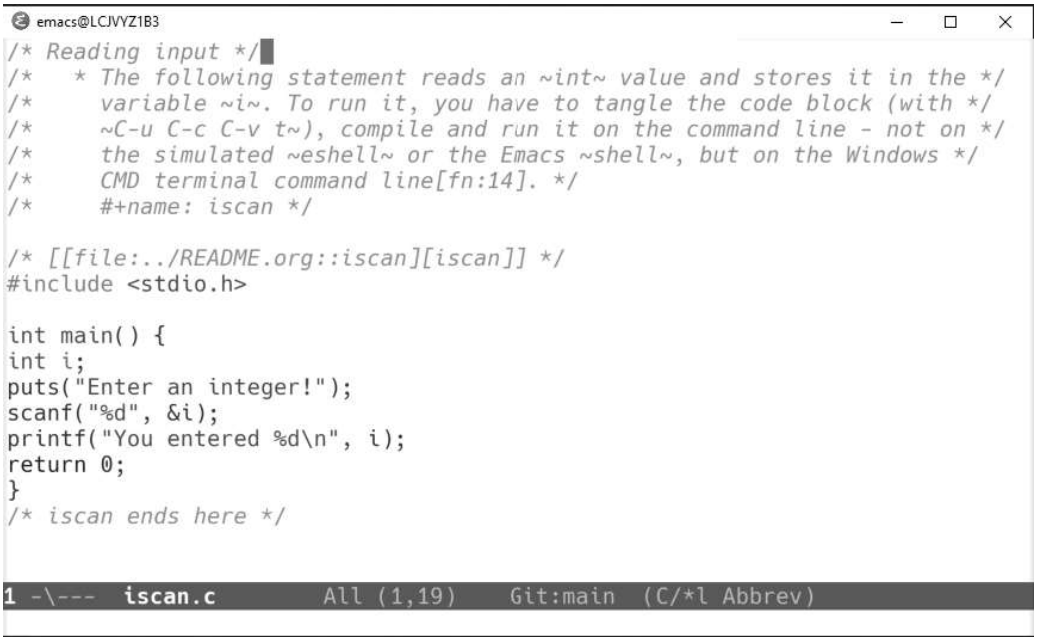


Figure 22: Tangled source code C file with comment

C Basics

CSC100 Introduction to programming in C/C++

- The following statement reads an `int` value and stores it in the variable `i`. To run it, you have to tangle the code block (with `C-u C-c C-v t`), compile and run it on the command line - not on the simulated `eshell` or the Emacs `shell`, but on the Windows CMD terminal command line[1].

```
int i;
puts("Enter an integer!");
scanf("%d", &i);
printf("You entered %d\n", i);
```

Footnotes:

[1] You could try and run it in Emacs. Can you explain the result?

Author: Marcus Birkenkrahe
Created: 2022-02-25 Fri 14:05

Figure 23: Documentation woven as HTML file in browser

```
#+name: iscan
#+begin_src C :tangle iscan.c :cmdline < input :results output
  int i;
  puts("Enter an integer!");
  scanf("%d", &i);
  printf("You entered %d\n", i);
#+end_src

#+RESULTS: iscan
: Enter an integer!
: You entered 1000
```

Figure 24: Executable C code block in Org-mode with results

19.6 Phone assignment solutions

- See GitHub folder assignments/

19.7 Program layout

- See GitHub folder 3_basics/

19.8 Next

- Solution to the `layout.c` challenge
- More on `scanf` and `printf` formatting (DIY notebook)
- Arithmetic expressions and operators (lecture)
- Lab session (in-class assignment notebook)
- Program assignment no. 6

20 Printf conversion - w9s20 (03/07/22)

20.1 To do

- [ ] Waking up in AppData/Roaming? Change your Emacs HOME now.
- [ ] Warming up: layout solution / NoppeLayout / NoppeLayoutModified
- [ ] Getting more deeply into formatting I/O with printf and scanf

20.2 Program Layout practice exercise

- Don't be stingy with text when it comes to documenting your work
- Documenting now will save you lots of time later

- Make use of these examples for the next program assignment

  See program examples NoppeLayout and NoppeLayoutModified.

20.3 Formatting I/O - printf

- Interactive notebook to code along (GDrive) - `io_printf_nb.org`

21 Scanf conversion - w921 (03/09/22)

21.1 Preparations for test 2 (Monday, 14-Mar)

- Test 2 will only cover questions from quiz 4-6 + new questions.

- You can find quiz 4-6 with solutions + feedback as PDF
- I will create an update of content Org files (in pdf/)

21.2 Warming up: `tprintf.c`

- Remember conversion specifiers m.pX
- New: a negative m left aligns a number: 123 as %-d => |123 |
- [ ]

  Guess the output before running 1

  - how many spaces m are reserved?
  - how many decimal places after the point?
  - is the number left or right aligned?

```
int i;
float x;

i = 40;
x = 839.21f;

printf("|%d|%5d|%-5d|%5.3d|\n", i, i, i, i);
printf("|%10.3f|%10.3e|%-10g|\n", x, x, x);
```

21.3 Formatting I/O - scanf

- Interactive notebook to code along (GDrive) - `io_scanf.org`
- Download the ZIP file and unpack it anywhere. All the files for the session are inside.

21.4 Programming assignment 6:

- Write a program `div3` that reads in three floating point numbers, adds the first two and divides the total by the third number. Test with these data:

```
(1610 + 2004) / 2365
```

- Bonus program for extra credit points: change the program to `div4`. This program should read in four floating point numbers. Add the first two together, then add the second two together. Divide your first sum by your second sum. Test with these data:

```
(1610 + 2004) / (2005 + 360)
```

- Submit in Schoology as an Org file with your name and `#+HONOR: pledged` in the header. The code blocks should be run and the `#+Result:` should show the correct answer.
- To get full points, submit the Org file with some comments outside of the code blocks - explaining what you do, what the code is about, or anything else really that you want to share[5]. Use NoppeLayoutModified.org as an inspiration.
- A comment could look this is: I absolutely, and unreservedly LOVE this class!
- This exercise originally came from Joyce (2018).

22 References

- For some of the programming projects, see: King (2011). C Programming. W Norton & Co.
- Joyce (2018). Numerics in C. Springer Apress.

Footnotes:

[1] This requires system admin privileges, which you only have on your own PC. In the computer lab, I have such principles, and as soon as I managed to install our tools, you can also use them on the lab equipment.

[2] I managed to install GCC on the lab computers and run it inside GNU Emacs. This is something that you should do at home with your own computer. I'm going to demonstrate the process in class and I will also make a short video showing how to do it (for Windows 10).

[3] In other Operating Systems but Windows, there is a fourth shell, the terminal (`M-x terminal`), and in Windows, you also have PowerShell. All of these are ways of interacting with the OS.

[4] GNU Emacs is an extensible text editor. Org-mode is a plain text environment in GNU Emacs for keeping notes, authoring documents, computational notebooks, and more. Tangling code means extracting (human readable) source code from an Org-mode file in order to run it. Weaving documentation means extracting a document from an Org-mode file in order to read it.

[5] It's a notebook, so you could for example write a paragraph reflecting on how easy or difficult you found this. Or you could add code blocks that didn't work and comment on why not.

Author: Marcus Birkenkrahe
Created: 2022-03-11 Fri 09:14
Validate