Lesson 1

Christian Schwarz, Jakob Krebs

November 24, 2019

Roadmap

Introduction

Make

Task

Introduction

Getting started

- we will use mostly linux
- all slides and examples will be available on github
  `https://github.com/jkrbs/c_lessons`
- all tasks can be send to us via e-mail and we will prvide feedback
  `c-lessons@deutschland.gmbh`
- information on our course will be available on `TODO`
- weekly lesons

development envirement

- you can use any editor of your choice
- you also can use an ide like vscode, clion, ...
- we will use a commandline, vim and gcc

gcc for Unix-based operating systems

Ubuntu / Debian:

```
$ sudo apt-get install gcc
```

Arch Linux:

```
$ sudo pacman -S gcc
```

Mac OS X:

```
$ brew install gcc
```

... and you're done ;-)

usage

Debug:

```
$ gcc -g -Wall -o output input.c
```

Release:

```
$ gcc -Wall -O2 -o output input1.c input2.c
$ strip output
```

Make

motivation

you have seen the gcc commands. there is a better solution

Makefile

```
CC := gcc
CFLAGS := -Wall
DFIAGS := $(CFLAGS) -g
RFLAGS := $(CLAFGS) -O2
EXE := our_binary_name
SRC := $(shell find src/ -iname '*.c')

# prevent make from treating targets as file names
.PHONY release clean debug

release:
$(CC) $(RFLAGS) -o $(EXE) $(SRC)

debug:
$(CC) $(DFLAGS) -o $(EXE) $(SRC)
```

usage

write your makefile in a file called “Makefile” in the root directory of your project

```
$ make # in the root directory of your project
```

Task

Task

Write a c program which finds prime numbers to a given maximum. And write a makefile to build it.

additional tasks

- make the program dynamically allocate memory and print primes until it is terminated
- use a faster prime finding algorithm like erastosthenes sieve
- write your output to a file given as a first commandline argument and display a progress on stdout

help for additional tasks: memory allocation

```
void* malloc(size_t size);
void free(void* ptr);
```

example:

```
#include <stdio.h>
int main() {
  int* foo = malloc(100* sizeof(int));
  for(int i = 0; i < 100; i++) {
    foo[i] = i;
  }
  free(foo);
  return 0;
}
```

help for additional tasks: file io

```
FILE* fopen(const char* path, const char* permissions);
size_t fwrite(const void* ptr, size_t size, size_t cnt,
FILE* stream);
size_t fread(void* ptr, size_t size, size_t cnt,
FILE* stream);
int fclose(FILE* filetoclose)
```

example:

```
int main() {
  FILE* f = fopen("foo.txt", "w");
  char* data = "lololol\n";
  size_t data_size = 8;
  assert(fwrite(data, data_size, 1, f) != data_size);
  fclose(f);
  return 0;
}
```