



Basic Course in C Programming

Pasi Sarolahti
3.6.2019

premo.aalto.fi/c19

Agenda

What is C, how it differs from Python?

Course arrangements

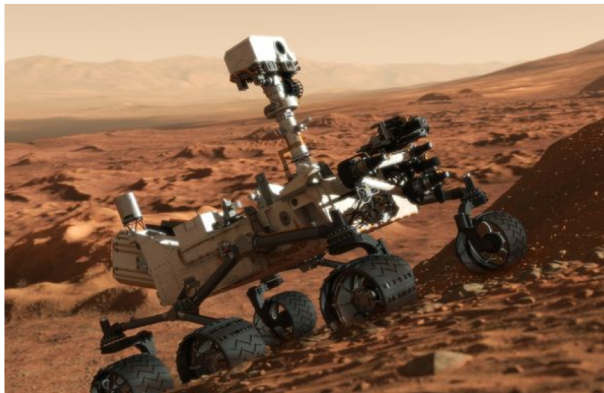
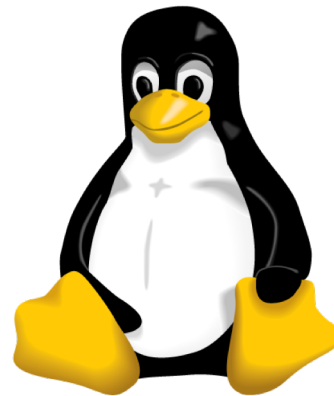
Programming tools and submission system

Example of solving a task

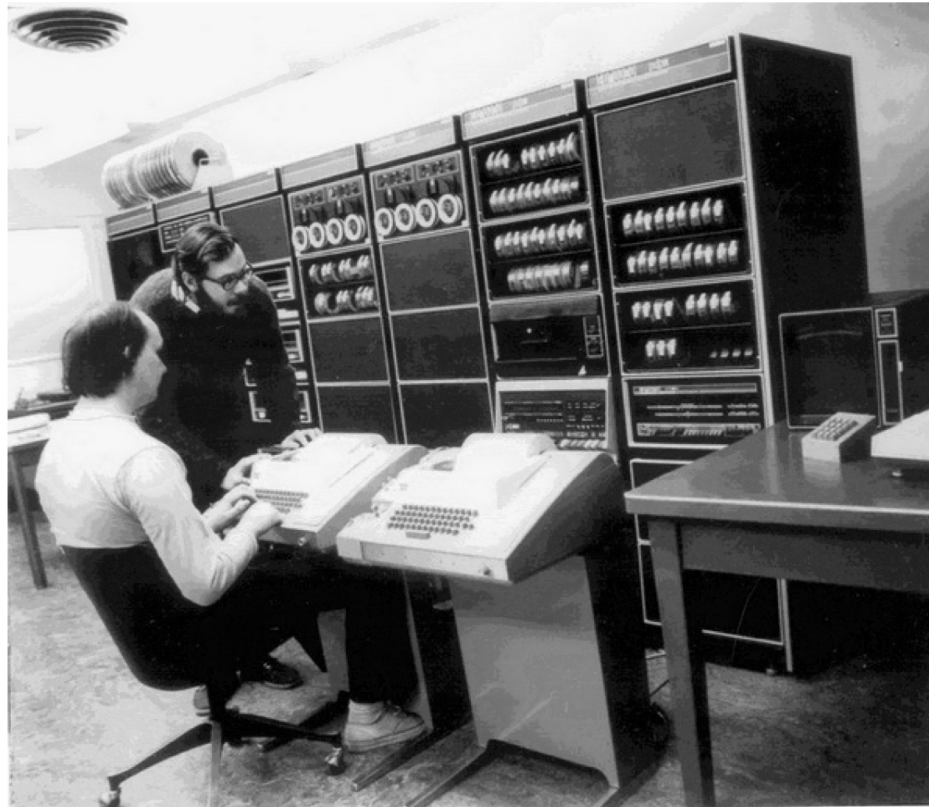
Objectives for the course

- "Tutustua C-kieleen ja oppia käyttämään sitä työkaluna muiden kielten rinnalla"
- "Olen kuullut, että kurssi on erittäin työläs ja haastava. Toivon, että onnistun tekemään tehtävät itse, ilman ulkopuolista apua"
- "Know enough about C to be able to learn C++ without too much trouble."
- "Tavoitteeni on oppia c-ohjelmoinnin perusteet erinomaisesti, koska uskon että c-ohjelmoinnista on hyötyä tulevaisuuden kannalta."
- "Haluan oppia miten, miksi ja missä tilanteissa C-kieltä voidaan hyödyntää."
- "Toivoisin, että tentti on järkevä, eikä hirveästi tarvitsisi "koodata paperille""
- "Toivottavasti on hauskaa. :D"
- "Odotan kurssilta runsaasti uutta ymmärrystä ohjelminnista yleisesti, etenkin syvällisempää ymmärtämistä tietokoneen toiminnasta ohjelmakoodin takana.

What C?



Ken Thompson & Dennis Ritchie



<http://www.computerhistory.org/fellowawards/hall/bios/Ken,Thompson/>

C vs Python

Python

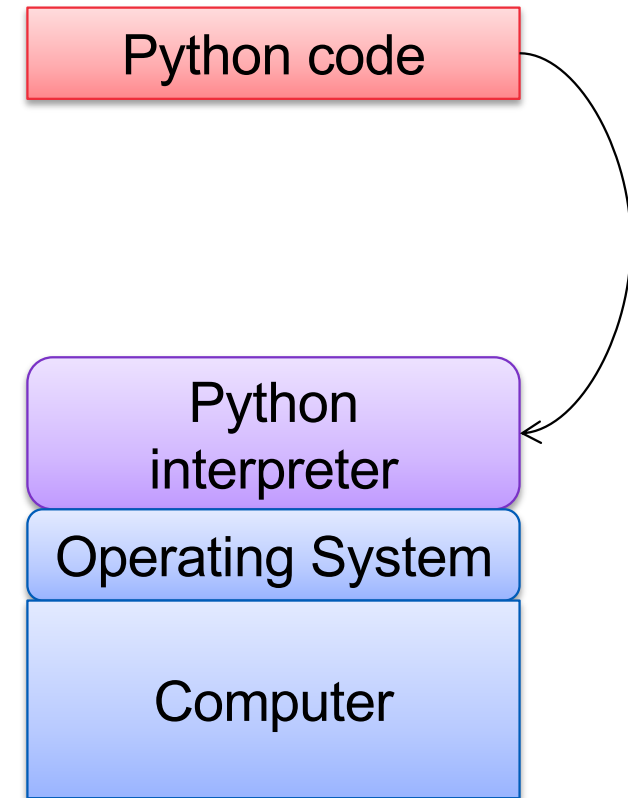
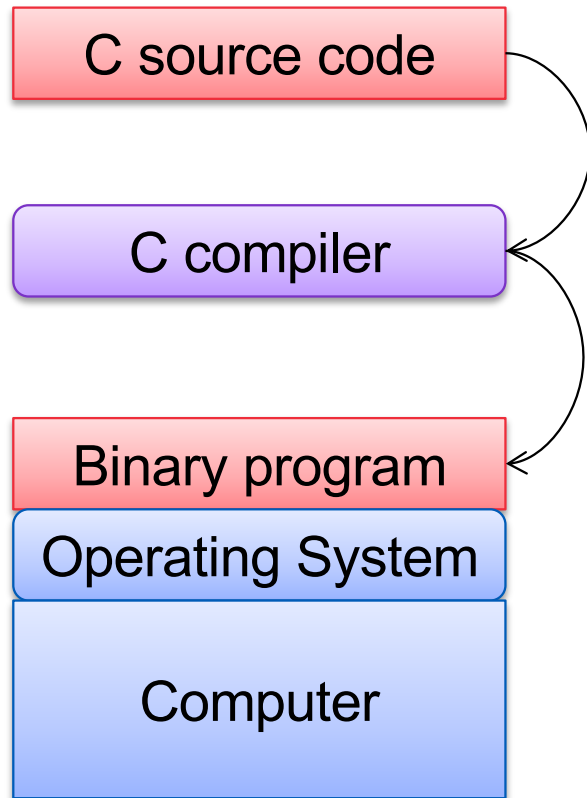
```
friends = ['john', 'pat', 'gary', 'michael']
for i, name in enumerate(friends):
    print "iteration {iteration} is {name}".format(iteration=i, name=name)
```

C

```
#include <stdio.h>

int main(void) {
    const char *friends[] = { "john", "pat", "gary", "michael" };
    for (int i = 0; i < 4; i++) {
        printf("iteration %d is %s\n", i, friends[i]);
    }
}
```

C vs. Python



C vs. Python differences

Formatting

- Technically, indentation and line endings don't matter in C

Variables

- Variables must always be declared before use, type cannot be changed afterwards

Syntax

- Semicolon terminates statement
- Braces used to indicate program blocks

Memory Management

C Properties

Small, simple low level language

Functionality in functions, data accessible through variables

- No classes, no objects

Static typing: type of variable determined at compile time

I/O, strings, etc. implemented as libraries

Learning objectives

Basics of C language, including:

- Basic structures, data types, syntax
- Pointers, memory management
- Binary operations

Ability to produce short C programs

Ability debug (i.e., find errors) from own and others' code

Basic understanding of computer architecture

Ways of working

One learns programming by programming

Many automatically assessed programming tasks

- Allows independent work

Material and exercises in network

No lectures

- Tue 6.8.: Information session on project and exam arrangements

Schedule (see details in MyCourses)

Assessment

40% Exercises

- Min. 500 points
- Min. 7 rounds must be completed with at least 50 points

30% Programming task

30% Exam

- At least 8 points required from max. 16
- Done using computer

Total grade will be calculated as weighed average from above

Programming test



- **Done using your own laptop, on a system booted from USB**
 - Test before actual exam
- **If USB does not work for you**
 - Some number of laptops for borrowing
 - Option to use exam aquarium

Exercises and Assistants

Protocol

- Short intro/demo in the beginning of exercise session
- Working on exercises, assistants help

We do not have ticket/queue system

- Assistants aim at distributing evenly in the class room, check through students in orderly way

Course Material

Discussion and questions

Preferred: Slack

- Join by: <https://aalto-c.slack.com/signup>
- No Aalto email? Contact course personnel using email (below)

You can share your solutions using a link from TIM

- Visibility: course personnel always, those students who have already solved the task. After deadline everyone.

Email: elec-a7100@aalto.fi

- Only for administrative issues

Producing a C Program

C programming steps

1. Write source code with text editor

- Source code files have .c suffix
- Larger software consists of several modules (several .c files)

2. Produce binary code

- **Precompiler:** remove comments, process headers, etc.
- **Compiler:** From source code to object files
- **Linker:** Combine object files into running program
- If there are warnings or errors, go back to 1 and fix

3. Test program

- If does not work correctly, go back to 1 and fix

Typical files in C programming

.c suffix indicates source code files

- There may be several in single program

.h suffix indicates header files

- Definitions of data types, data structures, etc.
- .c files refer to this using #include directive
- Commonly needed with libraries

Building the program

There are different compilers, such as “gcc” or “clang”

On terminal window (e.g. in Linux):

`gcc <lähdetiedosto.c>`

- Produces binary a.out file that can be executed

If there are several .c files, all are listed on the command line

- Typically on larger project “make” is used to make this easier
- Or one would use IDE, that does this automatically

Result of building

Everything fine: no output

Compile errors

- Compiler cannot produce executable file

Warnings

- Compiler is able to produce binary executable file
- However, code is very likely erroneous, therefore should be fixed

Look for line numbers in notifications to locate the error

Simple example

```
#include <stdio.h>
int main(void)
{
    /* The following line will print out some text */
    printf("hello, world\n");
}
```

- **#include**: include standard I/O functions in program (such as printf)
- **int main**: starts the main function that always begins program
- **/* .. */**: comment, can contain anything, ignored by compiler
- **printf**: function that outputs the string given in parameters

What happens to C program?

ASM

```
global _main
extern _printf
section .text
main:
    push    message
    call    _printf
    add     esp, 4
    ret
message:
    db 'hello, world', 10, 0
```

<https://excelwithbusiness.com/blog/say-hello-world-in-28-different-programming-languages/>

What happens to C program?

Konekieli

```
b8 21 0a 00 00 #moving "!\n" into eax
a3 0c 10 00 06 #moving eax into first memory location
b8 6f 72 6c 64 #moving "orld" into eax
a3 08 10 00 06 #moving eax into next memory location
b8 6f 2c 20 57 #moving "o, W" into eax
a3 04 10 00 06 #moving eax into next memory location
b8 48 65 6c 6c #moving "Hell" into eax
a3 00 10 00 06 #moving eax into next memory location
b9 00 10 00 06 #moving pointer to start of memory location into ecx
ba 10 00 00 00 #moving string size into edx
bb 01 00 00 00 #moving "stdout" number to ebx
b8 04 00 00 00 #moving "print out" syscall number to eax
cd 80          #calling the linux kernel to execute our print to stdout
b8 01 00 00 00 #moving "sys_exit" call number to eax
cd 80          #executing it via linux sys_call
```

<https://excelwithbusiness.com/blog/say-hello-world-in-28-different-programming-languages/>

About tools

Good old: command line + editor

- Editor: e.g. Atom, Kate, Emacs, Vi, Notepad++
- Compile program in terminal

Modern: Integrated Development Environment (IDE)

- Everything in single package
- Netbeans, Eclipse, Visual Studio, Xcode

Some instructions provided with course material

Have a nice course!