EL2310 – Scientific Programming

Lecture 6: Introduction to C

Carl Henrik Ek
(chek@csc.kth.se)

Royal Institute of Technology – KTH

Overview

Lecture 6: Introduction to C

- Roots of C
- Getting started with C
- Closer look at “Hello World”
- Homework

Schedule

- Last time (and before): MATLAB
- Today: Introduction to C - main part of this course
- Friday 16th of September: Deadline to submit your MATLAB project solutions.

The roots of C

- First compiler developed by Dennis Ritchie
- Was based on two languages:
  - BCPL, written by Martin Richards, and
  - B, written by Ken Thompson in 1970 for the first UNIX system
- Original C language was known as “K & R” C
  (Kernigan & Ritchie C (authors of the book))

ANSI C

- American National Standards Institute (ANSI) formed a committee
- to define “an unambigious and machine-independent definition of the language C”
- Committee formed 1983
- 1988 completed
- Resulted in ANSI C

The C language

- developed for UNIX
- the system and most programs written in C

The C language cont'd

- "System programming language"
  - *we need to relate things to the computer*
- compilers
- operating systems
- low level language (in contrast to MATLAB, for example)

A small sized language

- C does not provide functions for e.g.
  - manipulation of composite objects (arrays, lists, etc)
  - input/output
  - ...
- Most C implementations add such support

Typeing

Datatype:

- Classifies type of data, *integer*, *char*, ...
- Machine data-type

Operator:

- Interaction between objects, **+**,**-**, ...

Typeing

- Restrictions on interaction between data types
- Strongly, Weakly and Untyped
- Statically or Dynamically typed

Strongly typed languages are more likely to catch errors at compile time while weakly typed languages allow further flexibility.
type conversion,*overloading*

Typeing

Benjamin C. Pierce, author of Types and Programming Languages and Advanced Types and Programming Languages: *”I spent a few weeks... trying to sort out the terminology of ”strongly typed,” ”statically typed,” ”safe,” etc., and found it amazingly difficult.... The usage of these terms is so various as to render them almost useless.”*

- MATLAB “weakly” typed
- C “weakly” typed
- C++ confused

Learning C

- Practice!
- Practice!
- Practice!
- Practice!
- Practice!
- A very good idea: Define your own little project.

Steps to a running program

- **Write**
- **Compile**
- **Link**
- **Execute**

From: http://www.physics.drexel.edu/courses/Comp_Phys/General/C_basics/compile.html

Compiling the code

- parsing of the statements for syntax
- translation of the statements into machine language
- setting up the addresses of all variables
- optimization of the code (if desired)

Linking

- assembles the various routines produced during the compilation
- resolves missing calls to either language-specific libraries or system-wide functions

Optimization

- You can tell the compiler to optimize the code
- It is good to NOT do that until you know that the program works as it should
- Optimization changes the code internally for better efficiency
- Can typically specify different levels of optimization
- Optimization can in same cases introduce errors

The first program

- The Hello world program
- Typically the first program written in all languages

Hello world

- Input: nothing
- Output: prints “Hello world” on the screen

Hello world

```
#include <stdio.h>

main()
{
  printf(``Hello world\n'');
}
```

The gcc compiler

- GNU offers a freely available compiler
- called `gcc`

Compiling a program

- `gcc hello.c`
- If all goes well will produce a file called `a.out`

Running the program

- `./a.out`
- The prefix `./` instructs the system to run the program a.out in the current directory
- Just like in MATLAB there is a `PATH` variable that tells the system where to look for programs to run
- In Unix/Linux systems this `PATH` does normally not contain the current directory.

Compiler arguments

- Compiler takes many arguments
  - `-o <output filename>`
  - `-Wall output warning`
  - `-O, -O1, -O2, -O3` optimization level
  - `-c <filename.c>` only compile filename.c (not link)
  - `-lname` link to library called libname
  - `-L<directory>` tell linker where to find libraries
- For now let us focus on `-o`

Creating an executable cont'd

- To create executable `hello` from `hello.c`
- `gcc -o hello hello.c`

Analysis of the program

```
#include <stdio.h>

main()
{
  printf(``Hello world\n'');
}
```

- A C program consists of *functions* and *variables* (like in MATLAB)
- Functions are made up of statements (like in MATLAB)
- The executable starts in the function called `main`
- Each program has to have a `main`-function

Breakdown of the program

- Program starts with `#include <stdio.h>`
- Instructs the compiler to include information from the standard library for input and output (I/O)
- These lines are typically found at the top fo the file
- A function can have arguments
- In our case main was defined to have no arguments
- The statements for a function should be put between braces

The `printf` function

- `printf` is a command used to print output
- The argument is a string enclosed in double quotes
- Will see later that it can take more arguments
- The last character in the string is `\n` which is C style for the newline character
- There are many other “hidden” characters that you can get with a similar so called *escape sequence*
- `\t` will give you a tab for example

Homework until tomorrow

- Find a way to write, compile and run C programs!
- Start now
- Alternatives
  - Computers at CSC
  - Your own laptop
    - Linux
    - OS X
    - Virtual Machine (under Windows)

Editing files

- Simple text editors without IDE, like emacs, should be prefered.
- You can of course use your favorite text editor to edit the text files under windows.
- Avoid editors like words which adds lots of extra stuff to the file and make sure to save the file as text only.

Next Time

- Tomorrow Tuesday 13th 10-12 Q24
- Continue with C