MPCS 51040 – C Programming

Lecture 1

Dries Kimpe

September 26, 2016

Outline

- Useful Information
  - About me
  - Student Resources
  - Syllabus
- Some ground rules...
  - University Policy
  - Assignments
- Why C
- Hello World
  - Program
  - Environment
  - Secure Shell
  - Version Control
- Workflow
  - Overview
  - Preprocessing

Where you might have seen me. . .

Teaching C programming, Introduction to Computer Systems, Operating Systems, Parallel I/O, MPI (UChicago, UIC, KULeuven, UGent).

Previously Assistant Computer Scientist at Argonne National Laboratory (working on parallel and distributed storage).

Currently Senior Software Engineer at KCG (securities trading firm).

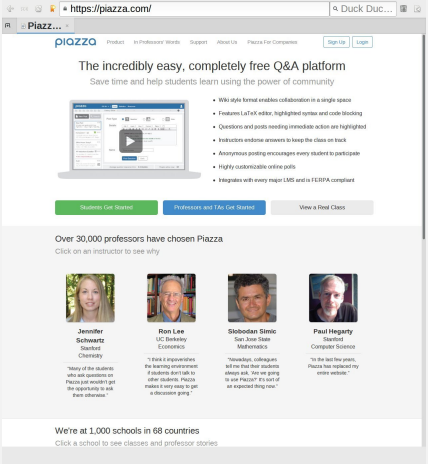
Community Service:

- Standard organizations: MPI, C++.
- Conference & workshops
- Journal review

Main research interests:

- High-Performance Computing and derived topics (networking, parallel computing, . . .)
- Operating Systems
- Software Development (from a process point of view)

Piazza



We will be using Piazza for:

- Class announcements
- Forum discussion

URL: `https://piazza.com/class/itj3cbhvhv83p6`

Homework & Other Assignments

Homework assignments will be distributed using the class git repository (more on this later). For handing in homework, you will be using your personal git repository.

- Clear timeline
- Can be used to show effort (recommended)
- Can handle multiple files easily

- Do not email homework to me or the TAs!
- We use automated tools to help with grading. Follow instructions exactly: filenames, upper/lower case, ...

Contact Information

In case the forum is not suitable don't hesitate to contact me (or TA) directly via email (or in person):

Lecturer Dries Kimpe <dries@cs.uchicago.edu >

TA Harish Naik(TBD)

Please use your official uchicago email address for class communication!

Office Hours

My office hours

- 1 hour before (appointment only) or after class.
- By appointment (both downtown or at the university).

TA Office Hours

To be determined; Please fill out .

Whenever possible email ahead of time (even if short notice)

Course Organization

Final Grade **approximate** weights

- Homeworks (and possibly final project): 60%–70%
- Quizzes: 30%–40%
- Class **and forum** participation: 5%–10%

Indication only!

Lectures

- 10 lectures
  - (possibly) optional exercise sessions (typically on Saturday morning)
- 8-9 homeworks, due **before** the next class
- 1-2 quizzes (≈90min each)
- 20-30 min break during each lecture
- Active participation expected

Syllabus

The syllabus can be found at `https://mpcs-courses.cs.uchicago.edu/2016-17/autumn/courses/51040`.

The syllabus is a guideline – schedule & coursework is subject to change based on class progress

UNIX Bootcamp is strongly recommended!
`https://csmasters.uchicago.edu/page/unix-bootcamp-0`

Review syllabus

Academic Honesty & Plagiarism

In brief, academic dishonesty (handing in someone else's work as your own, taking existing code and not citing its origin, etc.) will not be tolerated in this course. ***Penalties for academic dishonesty can range from failing the class to expulsion from the Masters Program.*** *Even so, collaboration between students is certainly allowed (and encouraged) as long as you don't hand someone else's work as your own. You can discuss the high-level aspects of assignments with other students, but you should never share your solution to an assignment with other students. If you have discussed parts of an assignment with someone else, then make sure to say so. If you consulted other sources, please make sure you cite these sources. Unless you have written an assignment entirely on your own with no outside assistance,* ***always err on the side of caution and disclose every source you have consulted.*** *If you have any questions regarding what would or would not be considered academic dishonesty in this course, please don't hesitate to ask the instructor.*

University Policy

Academic Honesty & Plagiarism Policy

Please review `https://csmasters.uchicago.edu/page/rules-and-policies` for a full description.

- Academic dishonesty is a serious offense and there will be consequences even for first time offenders.
- Discussion is generally fine if you solve the assignment on your own and you disclose who you discussed the assignment with.
- Always err on the side of caution.
- When in doubt, ask.

Unlawful Discrimination and Sexual Misconduct policy

University policy here.

Homework

Handing In

- In your personal git repository
- Make sure your files are pushed to the server (Check by recloning or via the web interface)
- Use file- & directorynames are as specified! (otherwise grading scripts might fail to find your work)
- Email and other methods are not accepted.

Late Policy

- Strict deadline will be enforced (but see next point).
- Exceptions can be made on a case-by-case basis.
- Incomplete/non-functioning will still yield points
  Commit early and often!

Do not wait until the last moment to request an exception!

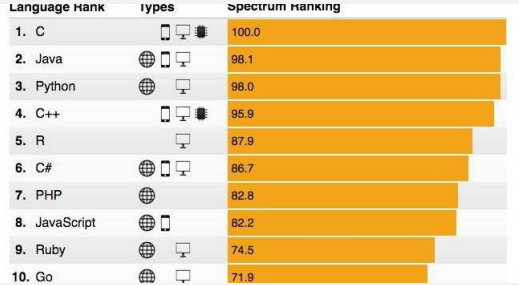
Why would you be interested in C?

There are many languages to choose from (and more are invented every year). Why would somebody be interested in learning C?

- Good basis for learning C++ (and other imperative languages)
- Because C doesn't offer many built-in language features, one develops a good understanding of how those features are implemented in other languages.
  - ...and of their performance implications...
- Not much happens behind the scenes. This offers full control over:
  - Memory usage (import for embedded systems **and** high-performance computing/big data)
  - Code generation (important for OS development)
  - Performance guarantees/constraints (real-time systems)
- C exposes a fair amount of the underlying hardware architecture
- Some areas traditionally still use C
  - Embedded system development
  - OS kernel programming
  - ...
- Many other languages are modeled (or based) on C
- Depending on which source is consulted, C one of the most popular languages

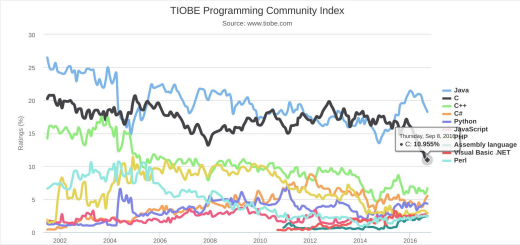
Recent examples

IEEE



Source: The Register "Plenty of Fish in the C", July 2016.
http://www.theregister.co.uk/2016/07/28/plenty_of_fish_in_the_c_ieee_finds_in_language_popularity_contest/

TIOBE



Source: TIOBE Progamming Community Index, September 2016.
http://www.tiobe.com/tiobe-index/

First C Program

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main (int argc, char ** args)
5 {
6     int ret = printf ("Hello, world!\n");
7     return EXIT_SUCCESS;
8 }
```

- 1,2: "module" concept
- 4: define (and declare) function
- 6: function call
- 6: note '\' *character(string) escape*
- 6: variables (static typing)
- declarations, definitions, statements and expressions...

Compiling: `gcc -Wall -pedantic -std=c11 -o hello hello.c`

Demo on linux.cs.uchicago.edu

Programming Environment

Recommended to work on linux.cs.uchicago.edu:

- That is where projects will be graded (unless explicitly noted otherwise)
- Can connect from any OS
- Available 24/24

However, if for some reason, you can't use linux.cs.uchicago.edu (for example due to no connectivity or firewall restrictions), *the recommended approach is to run Ubuntu 15.01 in a virtual machine*.

- Free Virtual Machine host software available: VirtualBox
- Will protect your machine against mistakes (using all memory . . .)
- Avoids system differences and allow reproducing issues

Regardless of which method you choose, it is *your* responsibility to test and validate your solution on linux.cs.uchicago.edu!

NOTE

The use of IDEs is strongly discouraged!

Virtualbox

- Homepage: `https://www.virtualbox.org/`
- Download:
  `https://www.virtualbox.org/wiki/Downloads`
  - Download platform package for your system
  - Download and install extension pack as well
  - No need for software developer kit
- Contact me to obtain VM image

Secure Shell

A protocol for connection to a remote terminal (where terminal is a device generating (text) output and taking keyboard input).

Unix-based systems

- Mostly provided as part of the base system
- Look for 'ssh' command

Windows/Mac Client

- Putty: `http://www.chiark.greenend.org.uk/~sgtatham/putty/`
- Many others available...

Since we will generally not be generating graphical output, a terminal is all we need to create, compile and execute the programs we will write for this course.

- Connect to linux.cs.uchicago.edu
- Demonstrate key-based authentication

- Keybased-authentication for linux: `https://www.digitalocean.com/community/tutorials/how-to-configure-ssh-key-based-authentication-on-a-linux-server`

Version Control

Why version control?

- Ubiquitous everywhere (except perhaps academia)
- Serves as backup and shows how code evolved
- Helps *collaborate* on code

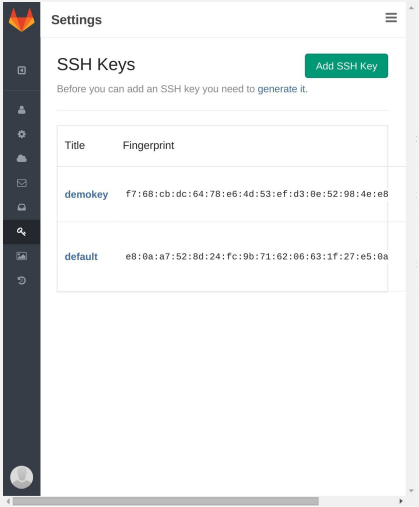
Important repositories

- GitLab server: `https://mit.cs.uchicago.edu`
- Class Repository: `https://mit.cs.uchicago.edu/mpcs51040-aut-16/mpcs51040-aut-16`
- Personal Repository: (differs for everybody)

Please let me know immediately if you do not have access to both the class repository or your personal repository!

Cloning class repository, index, adding file

Key-based authentication for mit.cs.uchicago.edu



- A git repository can be accessed using multiple tranport protocols.
  supported by `mit.cs.uchicago.edu`: HTTPS and SSH
- The authentication method of the transport is used
  $\Rightarrow$ we can reuse the public-key pair we created earlier

Installing the key:

- Login to `mit.cs.uchicago.edu`
- Click on the profile setting (upper right corner)
- Click on the key icon in the sidebar on the left
- Click 'add ssh key' and paste the contents of the *public* key file in the field.

An easy test:

- SSH (using your key for authentication) to `git@mit.cs.uchicago.edu` (i.e. username git)
- You should see 'Welcome to GitLab, <yourusername>

Git Clients & more. . .

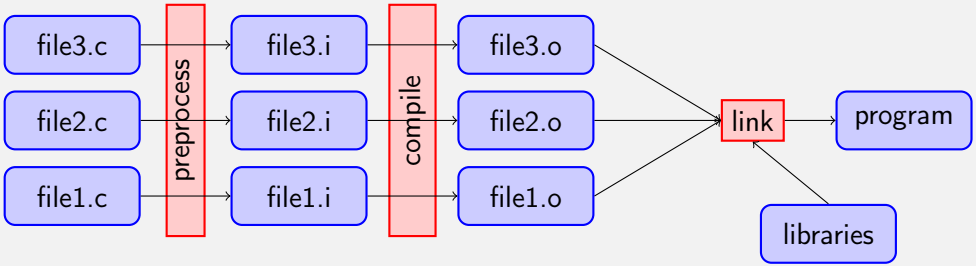
Setting up key-based authentication

Other useful resources

- Free git book: `http://git-scm.com/book/en/v2`
- GitLab documentation: `https://about.gitlab.com/documentation/`
- Write-up of using git+ssh on windows: `http://guides.beanstalkapp.com/version-control/git-on-windows.html`

From code to executable. . .

A processor can only execute a fixed set of binary instructions. How do we get from a set of files containing C code to an executable?



Every file is preprocessed and compiled *separately*

Preprocessor

Preprocessor?

- Completely different "language"
- Independent from C
- Its output is fed into compiler
- *text* manipulation language

What is it used for?

- Implement "module" concept
- Reduce repetition (for example constants)
- Conditional compilation

- The preprocessor *does not understand C!*
- Avoid side-effects (unpure functions)
- Precedence

Invoke using `cpp -pedantic -std=c11` or
`gcc -std=c11 -pedantic -E`

`#define`, `#ifdef`, `#include`, `max` problem

This week. . .

- Fill out class survey (`https://goo.gl/forms/e6s8akMWbRfDhEUz2`)
- Warm-up homework (will put on Piazza/emailed)
- Read chapter 1–3 of K&R book

Checklist

1. Verify access to class piazza page
2. Remember to read or forward your @uchicago.edu address
3. Verify `mit.cs.uchicago.edu` access to both the class repository (`https://mit.cs.uchicago.edu/mpcs51040-aut-16/mpcs51040-aut-16`) as well as to your personal repository (`https://mit.cs.uchicago.edu/mpcs51040-aut-16/yourcnetid`)
4. Fill out survey (`https://goo.gl/forms/e6s8akMWbRfDhEUz2`)
5. Complete TA office hour scheduling poll