CS-034

Introduction to System Programming

Thomas Doeppner

Pascal Van Hentenryck

Overview

- Why CS-034?
- Organization of the Class
- History of C

What is C?

- **A high-level assembly language**
  - all the nice things about assembly language
  - without most of the tedious aspects
- **What is tedious?**
  - Too verbose, low-level, unreadable
- **What is nice about assembly language?**
  - control about time, memory, and representation

What is C?

- Control of representation
  - bits, bytes, words
- Control of memory
  - dynamic allocation/deallocation
  - static and dynamic allocation
  - heap and stack allocation
- Low-level Manipulation
  - data structures
- Code Manipulation

What is C?

```
while (*s++ = *t++);
```

What is C used for?

- **Operating systems**
  - Unix, Linux, NT (Microsoft, Sun, Apple, IBM)
- **Databases, File systems, SCM**
  - Oracle, SAP, Network appliance
- **Networking, File systems**
  - CISCO,
- **Web**
  - Servers (Apache), Browsers (Mozilla), …
- **Runtime systems**
  - How do you write a Java system?
- **Others**
  - NLP, Robotics, Vision, Graphics, Optimization,

Why learn C? (by Joel Spolsky)

If you do not know C,

- you are programming by superstition

Why learn C? (by Joel Spolsky)

If you do not know C,

- you'll never be able to create efficient code in higher level languages

Why learn C? (by Joel Spolsky)

If you do not know C,

- You'll never be able to work on compilers and operating systems, which are some of the best programming jobs around.

- Add also
  - networking, embedded systems, databases, …

Why learn C? (by Joel Spolsky)

If you do not know C,

- You'll never be trusted to create architectures for large scale projects.

Why learn C? (by the CS-034 Staff)

If you do not know C,

- You'll never be able to read half the code ever written

Organization of the Class

- **Programming class**
  - Programming is the essence
- **Lectures**
  - Once a week: basic concepts
- **Labs**
  - Where the action really is!
- **Course load**
  - No exams,
  - No projects outside lab time

The Staff

- Head Teaching Assistants
  - Mark Johnson
  - Eric Tamura
- Teaching Assistants
  - Lucia Ballard
  - David Eustis
  - Todd Lipton
  - Stacy Wong

Organization again

Wednesday

- 3-4:30pm: Class

Thursday

- 4-6:00pm: Lab (3 TA's, A lab)
- 6-8:00pm: Lab (3 TA's, A lab)

Friday

- 3-5.00pm: Lab (3 TA's, B lab)

Monday

- 6-8.00pm: Lab (3 TA's, B lab)

Tuesday

- 7-8pm: Makeup Lab (1 rotating TA)

History of C

- 1969-1973
  - Development of C
  - Parallel to the development of Unix
- 1977-1979
  - Changes due to portability
  - K&R book
- Mid 1980s
  - Standardization: Ansi C

The Unix Operating System

- Ken Thompson (Bell Labs)
  - Took many ideas from Multics (MIT, Bell, GE)
  - Process, file structure, shell, text files
- Hardware environment
  - DEC PDP-7: 8K of 18-bit words
- First Unix System
  - PDP-7 assembly
  - Editor/assembler/shell/utilities (rm,cp,cat)
- Assembler
  - No loader/linker/libraries
  - One input file
  - One output file (a.out)

Unix and Programming Languages

- Pioneering concept (from Multics as well)
  - The OS must be written in a high-level language
- The B Programming Language (Thompson)
  - C without types
  - BCPL (Richards) in 8K
  - Written in TMG (McClure)
    - Language for writing recursive descent compilers
- Basic design decisions
  - Small
  - Close to the machine
  - Typeless
  - Libraries for I/O, system calls

The B Programming Language

- Overall Structure
  - Global declarations
  - Function declarations
  - Separate compilation
- Interesting/controversial decisions
  - Assignments
    - a = b;
  - Comments
    - /* this is a comment in B */
    - // this is a comment in BCPL

The B Programming Language

- **Bootstrapping**
  - Second version of B written in B itself

- **The B compiler did not generate assembly instructions**
  - Threaded code
  - Particular implementation of a virtual machine
  - In B, instructions acted on a stack

The B Programming Language

- **Constraints on B design**
  - The PDP-7 had 8K 18-bit words
  - The compiler must fit in the space
- **New features**
  - Takes space: compiler gets bigger
  - Reduces space: the compiler has more compact code
- **Examples**
  - `c++; x--;`
  - translation was smaller on the PDP-7
  - did not use, but was probably inspired, by auto-decrement memory cells.

The C Programming Language

- **1970: Unix is a promising system**
  - **PDP-11 with 24K of memory**
- **B Limitations**
  - **Untyped (or may be one type only)**
  - **Threaded code was too slow** 

The C Programming Language

- 1971: Birth of C (Dennis Ritchie)
- From B to C
  - Semantics of arrays & pointers
  - Typing and type composition
  - Structures
  - Preprocessor
- 1973: Modern C
  - Rewrite of Unix in C
- 1973-1980
  - Long, union, enumerated types
  - Unsigned int (why?)
  - casting

The C Programming Language

- **1982 Standardization**
  - **Portability!**
- **External functions**

```
extern int fact();
```

```
extern int fact(int n);
```

The C Programming Language

- **Learning C**
  - C is a simple language
  - Relatively easy to learn
- **Difficulties**
  - arrays and pointers: (not that bad)
  - syntax of type declarations and expressions
    - **reading may be hard at first**
- **Debugging**
  - memory errors
  - temporal disconnection between bug and symptom times.

The C++ Programming Language

- **Where?**
  - ATT Bell Labs again in the early 1980s (mostly)
- **Who?**
  - Bjarne Stroustrup
- **Why?**
  - Large event-based simulations
  - Simula-67 was too slow
    - Classes, inheritance, virtual functions
  - C++ for speed and functionalities

The C++ Programming Language

- **C with Classes (1980)**
  - Compiled to C
- **C++ (1983)**
  - virtual functions, operator overloading, references
- **Other features**
  - Templates: (from Ada, CLU, macros)
  - Exceptions: (from Ada, CLU, ML)
    - requires a real compiler, not a compiler to C
  - Multiple inheritance from classes (not interfaces)
  - Static member

Questions…