Systems Programming and Computer Architecture

Janis Hutz
https://janishutz.com

January 3, 2026

TITLE PAGE COMING SOON

"If you are using CMake to solve the exercises... First off, sorry that you like CMake"

- Timothy Roscoe, 2025

HS2025, ETHZ

Summary of the Lectures and Lecture Slides

Quotes

“An LLM is a lossy index over human statements“

- Professor Buhmann, Date unknown

“If you are using CMake to solve the exercises... First off, sorry that you like CMake“

“You can’t have a refrigerator behave like multiple refrigerators“

“Why is C++ called C++ and not ++C? It’s like you don’t get any value and then it’s incremented, which is true“

- Timothy Roscoe, 2025

Contents

1 The C Programming Language

I can clearly C why you'd want to use C. Already sorry in advance for all the bad C jokes that are going to be part of this section

`C` is a compiled, low-level programming language, lacking many features modern high-level programming languages offer, like Object Oriented programming, true Functional Programming, Garbage Collection, complex abstract datatypes and vectors, just to name a few. (It is possible to replicate these, more on this later).

On the other hand, it offers the ability to directly integrate assembly code into the `.c` files, as well as bit level data manipulation and extensive memory management options, again just to name a few.

This of course leads to `C` performing excellently and there are many programming languages who's compiler doesn't directly produce machine code or assembly, but instead optimized `C` code that is then compiled into machine code using a `C` compiler. This has a number of benefits, most notably that `C` compilers can produce very efficient assembly, as lots of effort is put into the `C` compilers by the hardware manufacturers.

2 x86 Assembly

3 Hardware