COP-3402 Systems Software

University of Central Florida
Paul Gazzillo

https://github.com/cop3402spring18/syllabus

People

Instructor:

Paul Gazzillo

@paul_gazzillo

pgazz.com

GTA:

Mesut Ozdag

UCF

About Me

- Assistant professor
  - Research and teaching
- Research interests
  - Software engineering: *analyzing configurable software*
  - Program analysis for security: *side-channel attacks*
  - Blockchain smart contracts: *concurrency and safety*
- Teaching interests
  - Programming languages
  - Program analysis
  - Software security

Research interest meetings:
- Starting 09/13
- Fridays 3pm in HEC-356

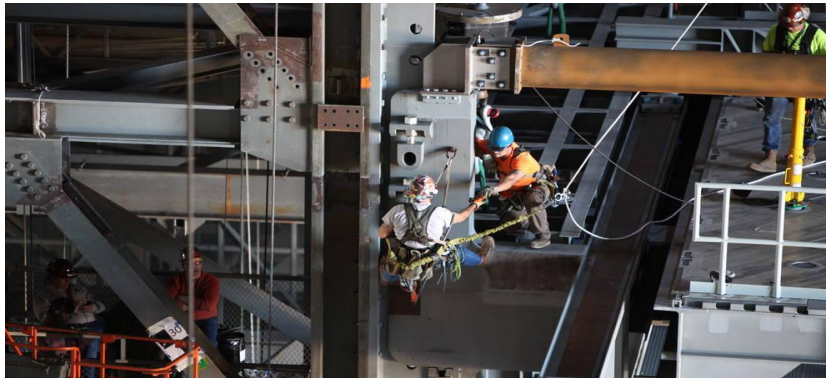
Overview

- Why should we study systems software?
- What is systems software?
- Class project: write a compiler
- Syllabus walkthrough: https://github.com/cop3402spring18/syllabus

Why Study Systems Software?

- Know your tools
- Be a better programmer
- Satisfy curiosity

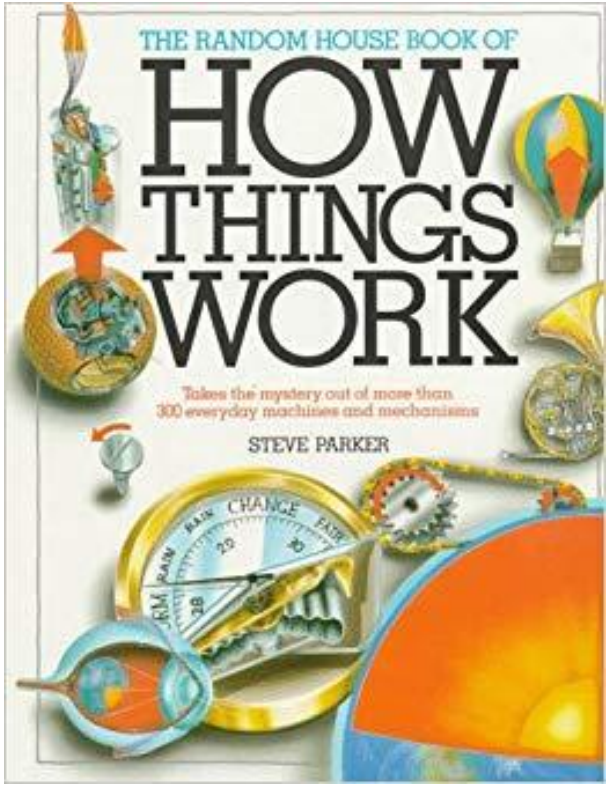
Know Your Tools



Be a Better Programmer

UCF

Satisfy Curiosity



What Is Systems Software?

Systems software is the set of programs that

1. support the operation of a computing machine
2. create an environment to run application software
3. support the development environment

Two Main Types of System Software

1. Development environment
    a. Editors
    b. Compilers
    c. Assemblers
    d. Linkers
    e. Debuggers
2. Run-time environment
    a. Operating system kernels
    b. Loaders
    c. Dynamic linkers
    d. Libraries

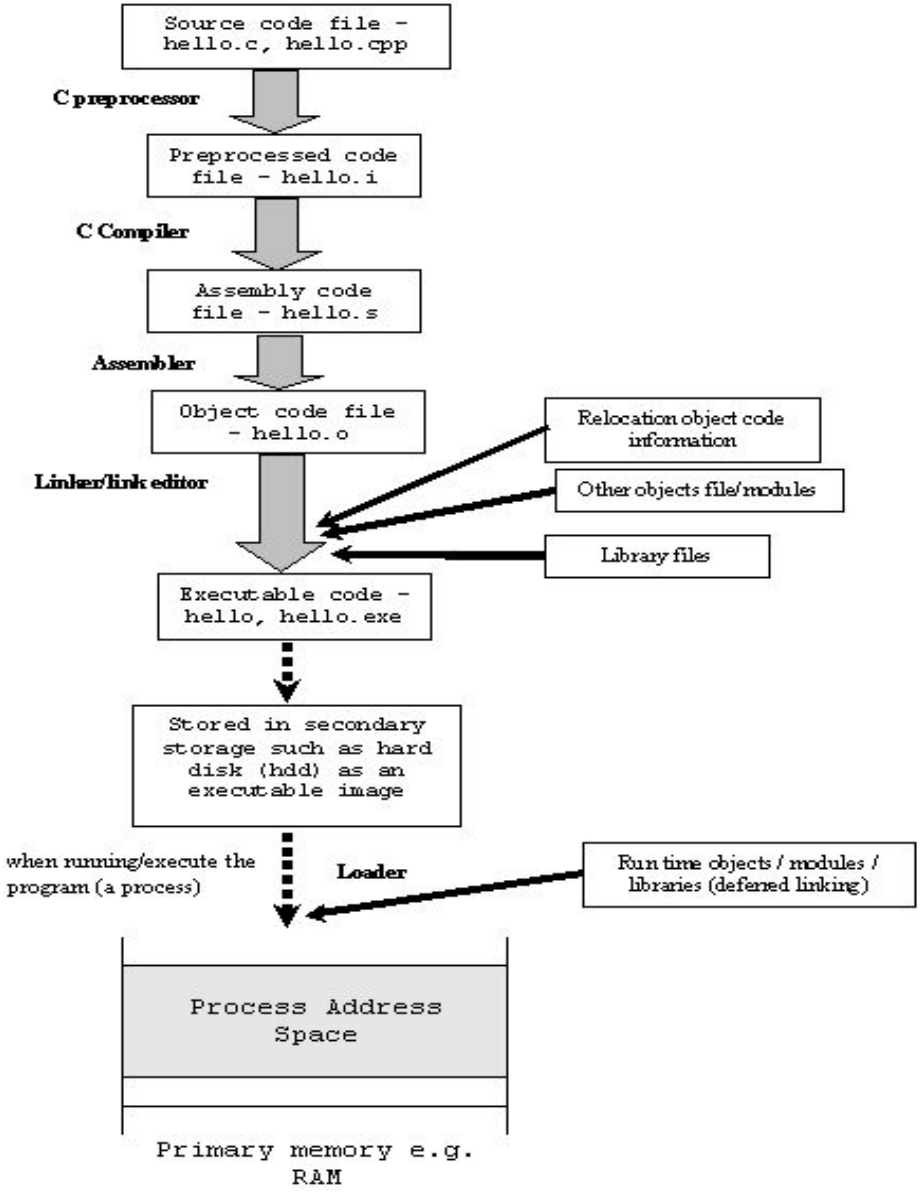
Development Environment

- ★ Editors
  - Creating and editing source files
- ★ Compilers
  - Translating source files to assembly
- ★ Assemblers
  - Translating assembly to object code (machine code)
- ★ Linkers
  - Collect multiple object files to produce a program
- ★ Debuggers
  - Examine state of a running program

Run-Time Environment

- ★ Operating system kernels
  - Abstract away hardware details and manages hardware access
- ★ Loader
  - Copies program into RAM and jumps to its first instruction
- ★ Libraries
  - Reusable code application software can link to e.g., `printf`
- ★ Dynamic linker
  - Loads libraries during execution instead of compile-time linking

From Source Code to a Running Program



Compilers Translate Source Code

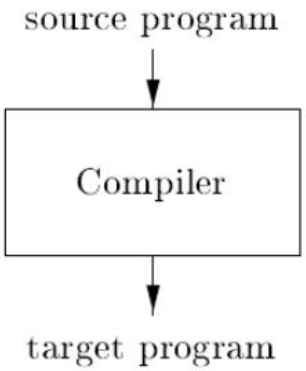


Figure 1.1: A compiler

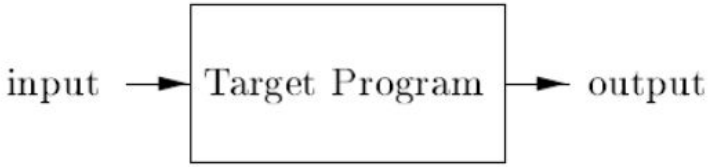


Figure 1.2: Running the target program

Interpreters Emulate a Machine

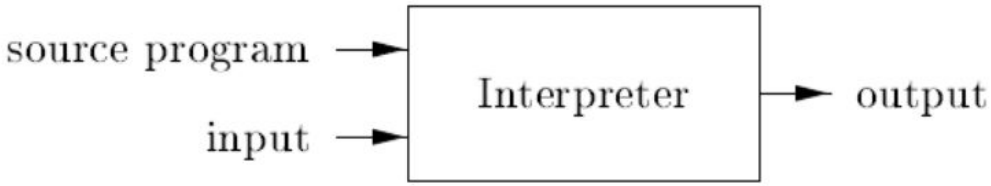


Figure 1.3: An interpreter

Class Project: Write a Compiler for SimpleC

- SimpleC is a very simple C-like language
- We will start with a complete, but restricted, language
- Each project will add new features to the language
- Five projects over the whole semester

Tooling for the Project

- Developing software requires systems software
  - Ulterior course motive: exposure to real-world tools
- Develop with prevalent, industry-standard tools
  - Automating your build with **make**
  - Tracking changes to source code with **git**
  - Running Linux on virtual machine with **vagrant** and **VirtualBox**
  - Targeting the **LLVM** compiler framework to generate binaries

Prepare Your Tools

https://github.com/cop3402fall19/syllabus/

Under **projects/** read the **README.md** file

1. Setup GitHub classrooms (**HW1 in webcourses**)
2. Setup vagrant and start your VM (**vagrant_setup.md**)
3. Setup your local git repository in the VM (**git.md**)

Next week: Project 0 (**project0.md)**

- Using the LLVM framework
- Linking and running LLVM intermediate code

Syllabus Walkthrough

https://github.com/cop3402fall19/syllabus