gcc Essentials

- Based in the Tutorial of
- Serguei A. Mokhov, mokhov@cs.concordia.ca
-

Contents

- Intro
- Options
- Examples

How a Program Works (cont'd.)

- Program must be copied from secondary (e.g. HDD) memory to main memory (RAM) each time CPU executes it
- CPU executes program in cycle:
  - Fetch: read the next instruction from memory into CPU
  - Decode: CPU decodes fetched instruction to determine which operation to perform
  - Execute: perform the operation

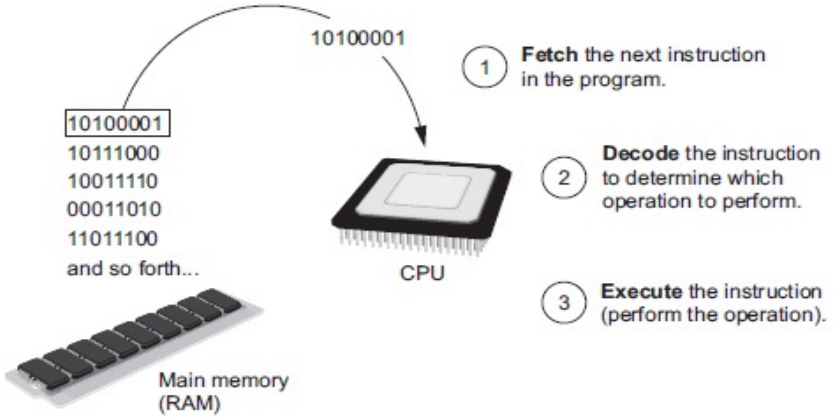


Figure 1-17 The fetch-decode-execute cycle

Computer Languages

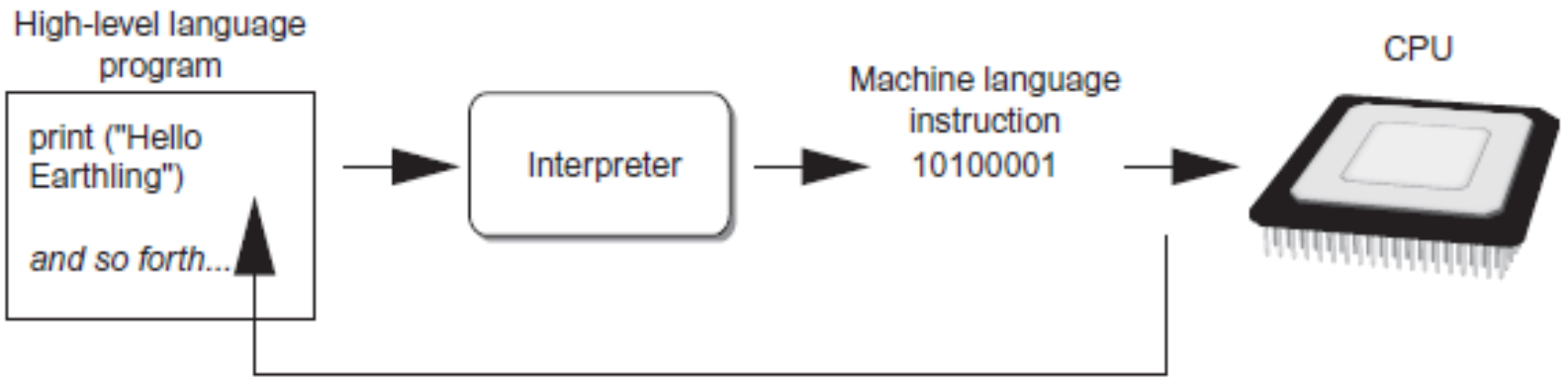
- Impractical for people to write in machine language
- Assembly language: uses short words (mnemonics) for instructions instead of binary numbers
  - Easier for programmers to work with
  - Assembler: translates assembly language to machine language for execution by CPU

- **Low-level language: close in nature to machine language**
  - Example: assembly language
- **High-Level language: allows simple creation of powerful and complex programs**
  - No need to know how CPU works or write large number of instructions
  - More intuitive to understand

Compilers and Interpreters

- Source code: statements written by programmer
  - Syntax error: prevents code from being translated
- Programs written in high-level languages must be translated into machine language to be executed
  - Compiler: translates high-level language program into separate machine language program
    - Machine language program can be executed at any time
  - Interpreter: translates and executes instructions in high-level language program
    - Used by Python language, Interprets one instruction at a time, No separate machine language program

Executing a high-level program with an interpreter

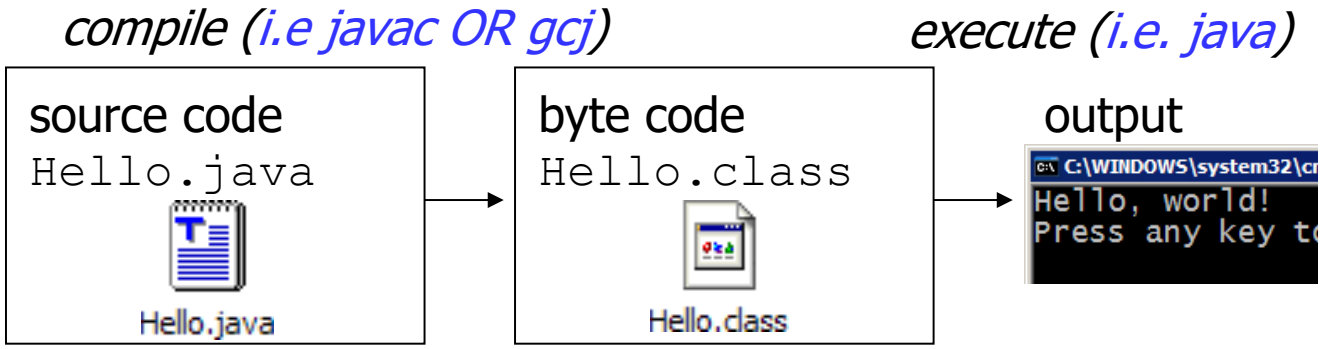


The interpreter translates each high-level instruction to its equivalent machine language instructions and immediately executes them.

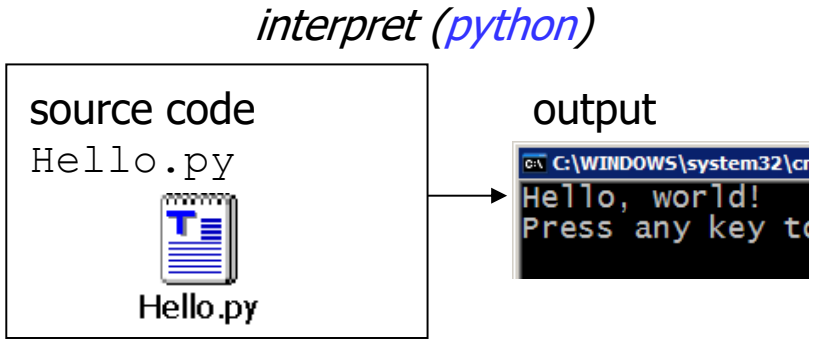
This process is repeated for each high-level instruction.

Compiling and interpreting

- Many languages require you to *compile* (translate) your program into a form that the machine understands.

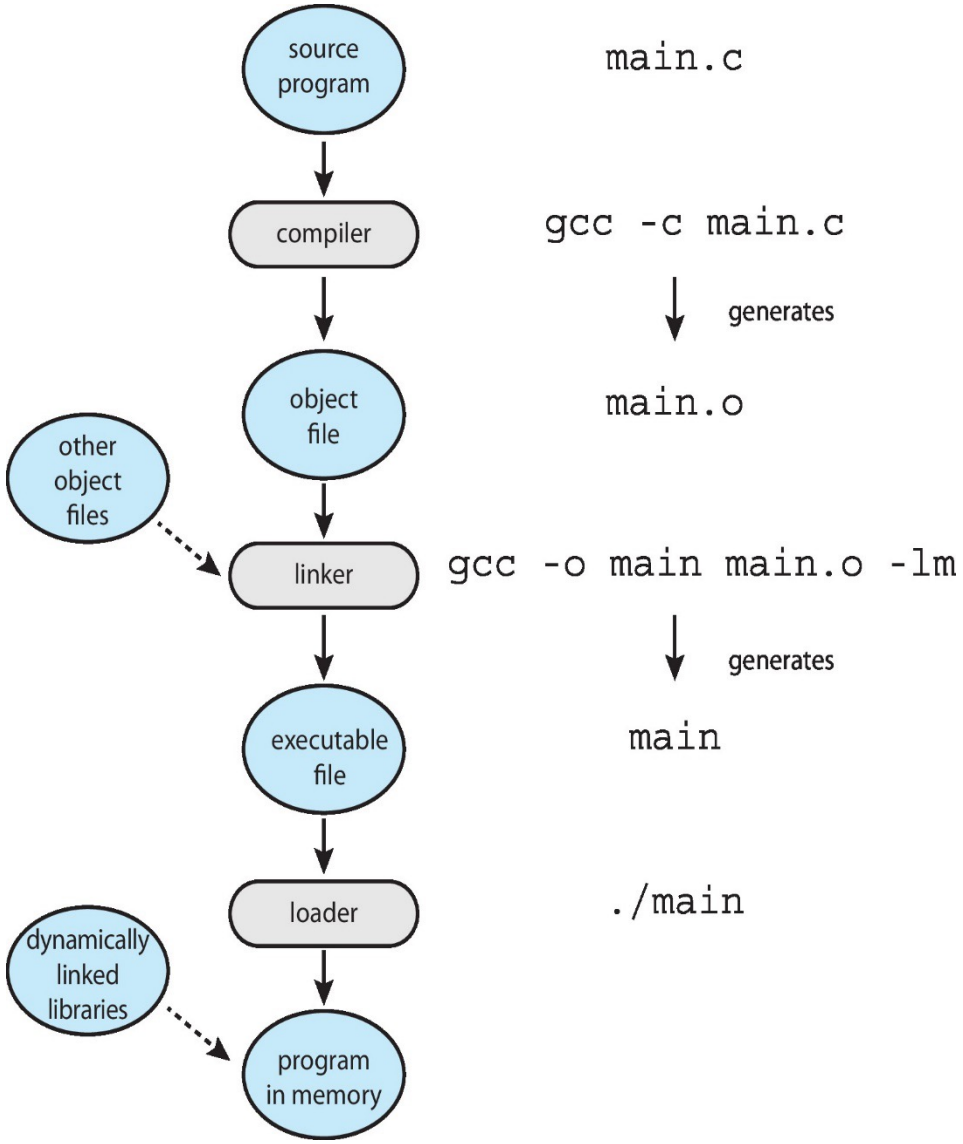


- Python is instead directly *interpreted* into machine instructions.

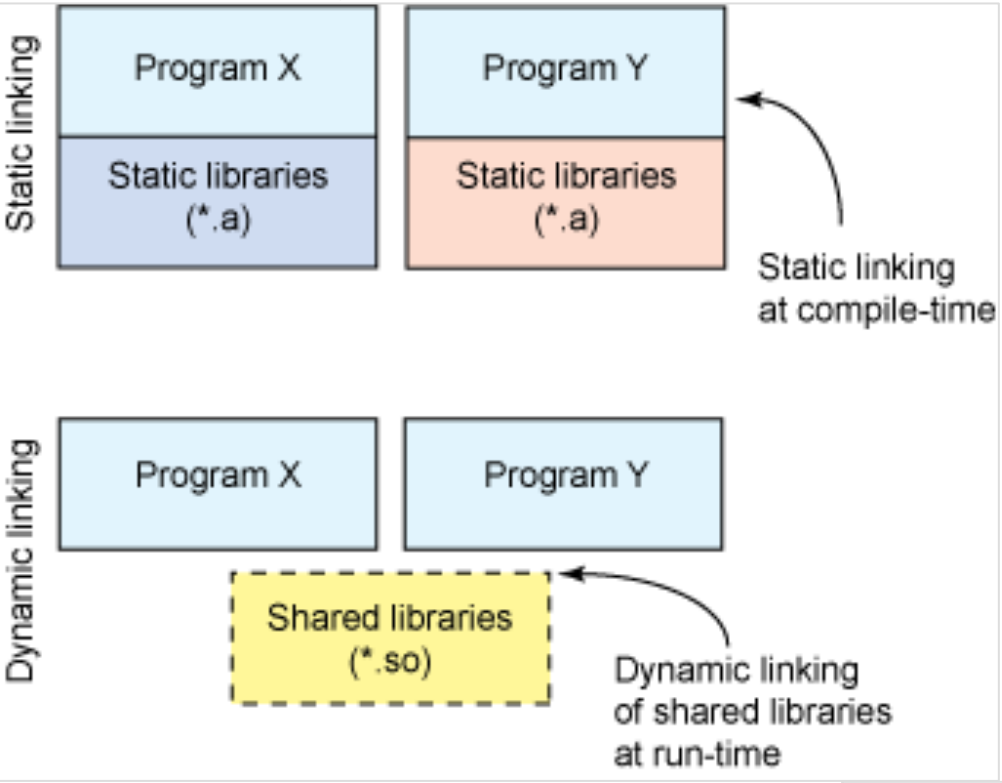


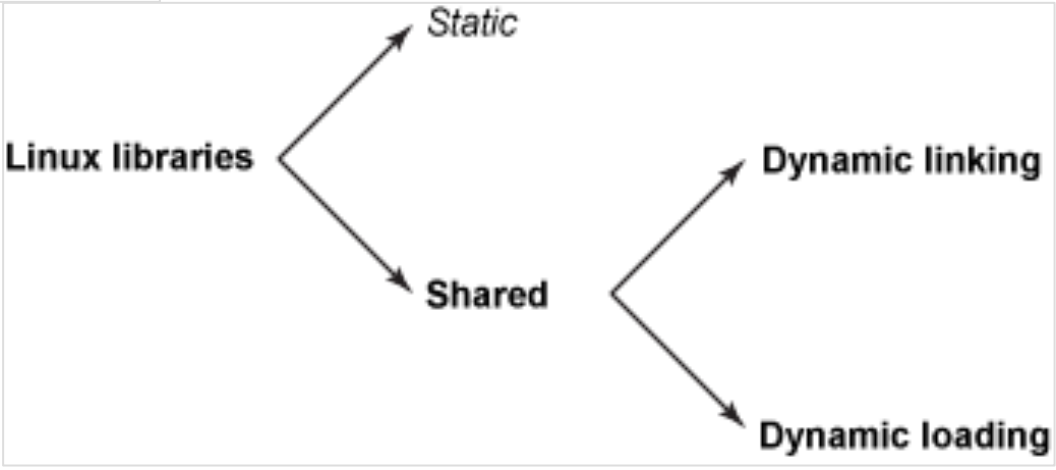
What is gcc?

- **gcc**
  - GNU C/C++ Compiler
  - Console-based compiler for Unix/Linux based platforms and others; can cross-compile code for various architectures
  - **gcc** for **C**; **g++** for **C++**
  - **gcc** performs all of these:
    - preprocessing
    - compilation
    - assembly and
    - linking
- As always: there is $ `man gcc`



Dynamic linking within Linux





Web: Anatomy of Linux dynamic libraries

`gcc` Options

- There are zillions of them, but there are some the most often used ones:
  - To compile: **`-c`**
  - Specify output filename: **`-o <filename>`**
  - Include debugging symbols: **`-g`**
  - GDB friendly output: **`-ggdb`**
  - Show all (most) warnings: **`-Wall`**
  - Be stubborn about standards: **`-ansi`** and **`-pedantic`**
  - Optimizations: **`-O`**, **`-O*`**

Options: `-c`

- **`gcc -c :`**
  - performs compilation and assembly of the source file *without* linking.
- The output are usually object code files, `*.o`;
  - they can *later* be linked and form the desired executables.
- Generates one object file per source file keeping the same prefix (*`before .`*) of the filename.

Options: `-o <filename>`

- **`gcc -o :`**
  - Places resulting file into the filename specified instead of the default one; *what is the name of this default ?* ☺
- Can be used with any generated files
  - object, executables, assembly, etc.
- If you have the file called **`source.c`**; the defaults are:
  - **`source.o`** if `-c` was specified
  - **`a.out`** if executable
- These can be overridden with the **`-o`** option.

Options: `-g`

- **`gcc -g :`**
  - Includes debugging info in the generated object code. This info can later be used in **`gdb`**.
- **`gcc`** allows to use **`-g`** with the optimization turned on (**`-O`**) in case there is a need to debug or trace the optimized code.

Options: `-ggdb`

- **`gcc -ggdb :`**
  - In addition to **`-g`** produces the most GDB-friendly output if enabled.

Options: `-Wall`

- **`gcc -Wall :`**
  - Shows most of the warnings related to possibly incorrect code.
- **`-Wall`** is a combination of a large common set of the **`-W`** options together. These typically **include**:
  - ***unused*** variables
  - possibly ***uninitialized*** variables when in use for the first time
  - ***defaulting*** return types
  - missing braces and parentheses in certain context that make it ambiguous
- Always a recommended option to save you from some "hidden" bugs.
- Try always using it and avoid having those warnings ☺

Options: `-ansi` and `-pedantic`

- For those who are picky about ***standard*** compliance.
- **`-ansi`** ensures that the code **complies** with ANSI C standard;
- **`-pedantic`** makes it even more strict.
- These options can be quite annoying for those who don't know C well since **`gcc`** will refuse to compile any code that does not follow the ANSI C standard, which otherwise it has no problems with.

Options: `-O`, `-O1`, `-O2`, `-O3`, `-O0`, `-Os`

- Various levels of optimization of the code
- **`-O1`** to **`-O3`** are various degrees of optimization targeted for speed (*performance*)
- If **`-O`** is added, then the code size is considered
- **`-O0`** means "*no optimization*"
- **`-Os`** targets generated code size
  - forces not to use optimizations resulting in bigger code.

Options: -I

- Tells **gcc** where to look for include files (**.h**).
- Can be any number of these.
- Usually needed when including headers from various-depth directories in non-standard places without necessity specifying these directories within the .c files themselves,
- e.g.:

```
#include "myheader.h" vs.
#include "../foo/bar/myheader.h"
```

For Your Assignments

- For your assignments, I'd strongly suggest to always include **`-Wall`** and **`-g`**.
- Optionally, you can try to use **`-ansi`** and **`-pedantic`**, which is a bonus thing towards your grade.
- Do not use any optimization options.
- You won't need probably the rest as well.

Quiz Time

- Observe the assigned quiz in the site.