Programming Languages

Janyl Jumadinova

September 8, 2020
Compiled? Interpreted? ...
Compiled? Interpreted? ... The language is designed, then the developer(s) implementing it decide whether to compile it or interpret it.
Compiled? Interpreted? ... The language is designed, then the developer(s) implementing it decide whether to compile it or interpret it.

**Pure Compilation**

Programs are translated into machine language.
Compiled? Interpreted? ... The language is designed, then the developer(s) implementing it decide whether to compile it or interpret it.

**Pure Compilation**
Programs are translated into machine language.

**Pure Interpretation**
Programs are interpreted by another program known as an interpreter.
The language is designed, then the developer(s) implementing it decide whether to compile it or interpret it.

**Compiled? Interpreted? ...**

**Pure Compilation**
Programs are translated into machine language.

**Pure Interpretation**
Programs are interpreted by another program known as an interpreter.

**Hybrid Implementation Systems**
A compromise between compilers and pure interpreters.
Pure Compilation

The compiler translates the high-level source program into an equivalent target program (typically in machine language), and then goes away.
Example

C (compiled)

$ gcc hello.c -o hello /*Compile source hello.c into tar*/
$ ls
hello hello.c
$ ./hello /* Execute target program ‘‘hello’’*/
Hello World
Pure Interpretation

- Interpreter stays around for the execution of the program.
- Interpreter is the point of control during execution.
Examples

Python (in interactive mode – interpreted)

$ python
>>> x = ‘Hello, world! ’
>>> y = 4
>>> y*x
‘Hello, world! Hello, world! Hello, world! Hello, world! Hello, world! ’
>>> x+y

Traceback (most recent call last):
  File ‘<stdin>”, line 1, in <module>
TypeError: cannot concatenate ‘str’ and ‘int’ objects
Example

REPL: “Read-Eval-Print-Loop”

**User repeatedly types in expressions that are immediately interpreted**

**Examples:** Python (previous slide); bash (command shell in Unix):

```
$ ls *
hello hello.c
$ cat hello.c
#include <stdio.h>
int main() {
    printf("Hello World\n");
}
```
Most language implementations include a mixture of both compilation and interpretation.
Common case is compilation or simple pre-processing, followed by interpretation.

Most language implementations include a mixture of both compilation and interpretation.
Java

$ javac Hello.java
// javac compiler produces byte code ‘.class’ file
$ ls
Hello.class Hello.java
$ java Hello Hello, world!

The Java Virtual Machine, or JVM (a “just-in-time” compiler), converts bytecode “on the fly” into machine code. (Opinions vary on whether to call this an interpreter!)
Compilation vs. Interpretation

**Interpretation:**
- Greater flexibility.
- Better diagnostics (error messages), easier to debug.
- E.g., in a REPL, programmer can decide what to do next based on output seen so far.
Compilation vs. Interpretation

Interpretation:

- Greater flexibility.
- Better diagnostics (error messages), easier to debug.
- E.g., in a REPL, programmer can decide what to do next based on output seen so far.

Compilation:

- Better performance.
- Privacy of the original code.
- Can consider whole program at once, optimize based on things like “remove unnecessary commands from loop body”.
- Not cross-platform.
Compilation vs. Interpretation

- Many, many variations, e.g., multiple compilation steps, compilers for interpreted languages, etc.
- The output of a compiler does not have to be “machine language’.”
- A lot of work to speed up interpreters.
Most Important Steps in Compilation

- Character stream
  - Scanner (lexical analysis)

- Token stream
  - Parser (syntax analysis)

- Parse tree
  - Semantic analysis and intermediate code generation

- Abstract syntax tree or other intermediate form
  - Machine-independent code improvement (optional)

- Modified intermediate form
  - Target code generation

- Target language (e.g., assembler)
  - Machine-specific code improvement (optional)

- Modified target language
Other Steps Possible

- Pre-processing prior to or in conjunction with lexical analysis.
- Final machine-specific optimization step.