CS201 - Programming Languages

Compiler Vs Interpreter

Aravind Mohan

Allegheny College

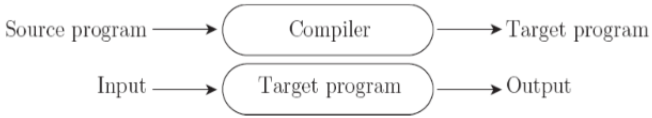
September 1, 2021

Compilation vs. interpretation

- Not opposites
- Not a clear-cut distinction

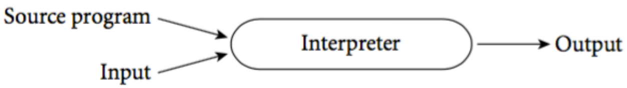
Pure Compilation

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



Pure Interpretation

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



Examples

C (compiled)

```
$ gcc hello.c -o hello
/*Compile source hello.c into tar*/
$ ls
hello hello.c
$ ./hello
/* Execute target program ''hello''*/
Hello World
```

Examples

Python (in interactive mode - interpreted)

```
$ python
>>> x = ``Hello, world! ''
>>> y = 4
>>> y*x
'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
```

Examples

REPL: “Read-Eval-Print-Loop”

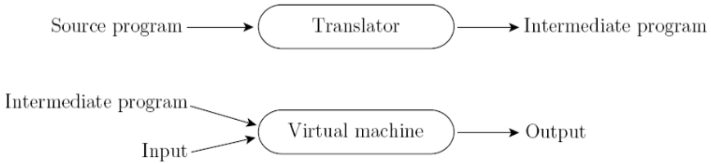
User repeatedly types in expressions that are immediately interpreted

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");
  }
```

Compilation vs. Interpretation

- Common case is compilation or simple pre-processing, followed by interpretation.
- Most language implementations include a mixture of both compilation and interpretation.



Examples

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!)

Interpretation

Interpretation:

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

Compilation:

- Better performance
- Can consider whole program at once, optimize based on things like “remove unnecessary commands from loop body”

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”.

Most Important Steps in Compilation

- Lexical analysis (scanning)
- Syntax analysis (parsing)
- Semantic analysis
- Intermediate code generation
- Optimization (usually machine-independent)
- Final code generation

Other Steps Possible

- Preprocessing prior to or in conjuction with lexical analysis
- Final machine-specific optimization step

Reading Assignment

PLP Chapter 01, Section 1.4, 1.6