COMP110: Principles of Computing

9: Compilers and interpreters

Learning outcomes

- Outcome 1
- Outcome 2
- Outcome 3

FALMOUTH
UNIVERSITY

How programs are executed

Executing programs

- CPUs execute **machine code**

Executing programs

- CPUs execute **machine code**
- Programs must be **translated** into machine code for execution

Executing programs

- CPUs execute **machine code**
- Programs must be **translated** into machine code for execution
- There are three main ways of doing this:

Executing programs

- CPUs execute **machine code**
- Programs must be **translated** into machine code for execution
- There are three main ways of doing this:
  - An **interpreter** is an application which reads the program source code and executes it directly

Executing programs

- CPUs execute **machine code**
- Programs must be **translated** into machine code for execution
- There are three main ways of doing this:
  - An **interpreter** is an application which reads the program source code and executes it directly
  - An **ahead-of-time (AOT) compiler**, often just called a **compiler**, is an application which converts the program source code into executable machine code

Executing programs

- CPUs execute **machine code**
- Programs must be **translated** into machine code for execution
- There are three main ways of doing this:
  - An **interpreter** is an application which reads the program source code and executes it directly
  - An **ahead-of-time (AOT) compiler**, often just called a **compiler**, is an application which converts the program source code into executable machine code
  - A **just-in-time (JIT) compiler** is halfway between the two — it compiles the program on-the-fly at runtime

Examples

Interpreted:

- Python
- Lua
- JavaScript (in old web browsers)
- Bespoke scripting languages

Examples

Interpreted:

- Python
- Lua
- JavaScript (in old web browsers)
- Bespoke scripting languages

Compiled:

- C
- C++
- Swift

Examples

Interpreted:

- Python
- Lua
- JavaScript (in old web browsers)
- Bespoke scripting languages

Compiled:

- C
- C++
- Swift

JIT compiled:

- Java
- C#
- JavaScript (in modern web browsers)
- Jython

Interpreter vs compiler

- Run-time efficiency: compiler > interpreter

Interpreter vs compiler

- Run-time efficiency: compiler > interpreter
  - The compiler translates the program **in advance**, on the developer's machine

Interpreter vs compiler

- Run-time efficiency: compiler > interpreter
  - The compiler translates the program **in advance**, on the developer's machine
  - The interpreter translates the program **at runtime**, on the user's machine — this takes extra time

Interpreter vs compiler

- Portability: compiler $<$ interpreter

Interpreter vs compiler

- Portability: compiler $<$ interpreter
  - A compiled program can only run on the operating system and CPU architecture it was compiled for

Interpreter vs compiler

- Portability: compiler $<$ interpreter
  - A compiled program can only run on the operating system and CPU architecture it was compiled for
  - An interpreted program can run on any machine, as long as a suitable interpreter is available

Interpreter vs compiler

- Dynamic language features: compiler $<$ interpreter

Interpreter vs compiler

- Dynamic language features: compiler $<$ interpreter
  - The interpreter is already on the end user's machine, so programs can use it e.g. to dynamically generate and execute new code

Interpreter vs compiler

- Dynamic language features: compiler $<$ interpreter
  - The interpreter is already on the end user's machine, so programs can use it e.g. to dynamically generate and execute new code
  - The AOT compiler is not generally on the end user's machine, so this is more difficult

Interpreter vs compiler

- JIT compilers have similar pros/cons to interpreters, but are generally faster (e.g. code inside a loop only needs to be translated once, then can be executed many times)

Interpreter vs compiler

- JIT compilers have similar pros/cons to interpreters, but are generally faster (e.g. code inside a loop only needs to be translated once, then can be executed many times)
- For games, run-time efficiency is usually much more important than portability

Virtual machines

Virtual machines

- Many modern interpreters and JIT compilers translate programs into **bytecode**

Virtual machines

- Many modern interpreters and JIT compilers translate programs into **bytecode**
- Bytecode is essentially machine code for a **virtual machine (VM)**

Virtual machines

- Many modern interpreters and JIT compilers translate programs into **bytecode**
- Bytecode is essentially machine code for a **virtual machine (VM)**
- Translation from source code to bytecode can be done ahead of time

Virtual machines

- Many modern interpreters and JIT compilers translate programs into **bytecode**
- Bytecode is essentially machine code for a **virtual machine (VM)**
- Translation from source code to bytecode can be done ahead of time
- At runtime, translate the bytecode (by interpretation or JIT compilation) into machine code for the physical machine

Virtual machines

- Many modern interpreters and JIT compilers translate programs into **bytecode**
- Bytecode is essentially machine code for a **virtual machine (VM)**
- Translation from source code to bytecode can be done ahead of time
- At runtime, translate the bytecode (by interpretation or JIT compilation) into machine code for the physical machine
- E.g. a Java JAR file, a .NET executable, a Python .pyc or .pyo file all contain bytecode for their respective VMs