INDIANA UNIVERSITY (HTTP://WWW.IU.EDU)

ARCHIVED: What is the difference between a compiled and an interpreted program?

Most programs are written in a high-level language such as C, Perl (afhp), or Java. Just as a human language makes it easy for people to communicate with one another, so computer languages simplify the job of telling a computer what to do. However, because a computer only understands numbers, talking to one is like speaking to someone with whom you don't share a language. You need a translator to properly communicate, and that's what interpreters and compilers do.

The difference between an interpreted and a compiled language lies in the result of the process of interpreting or compiling. An interpreter produces a result from a program, while a compiler produces a program written in assembly language. The assembler of architecture then turns the resulting program into binary code. Assembly language varies for each individual computer, depending upon its architecture. Consequently, compiled programs can only run on computers that have the same architecture as the computer on which they were compiled.

A compiled program is not human readable, but instead is in an architecture-specific machine language. Creating a compiled program requires several steps. First, the programmer, using a development tool or even a simple text editor, writes the source code in a chosen computer language. If the program is complex, pieces of it may be spread across several files. The programmer then compiles the program, sorting and linking the modules and translating it all into machine code that the computer understands.

Because different kinds of computers do not speak each others' machine languages, a compiled program will only work on the platform it was designed for. For example, a program written for HP-UX normally will not work on a Mac OS computer or a computer running Solaris. Despite this drawback, compiled programs are faster than those that must be run through an interpreter. Also, it is often possible to recompile the program so that it will run on different platforms. Examples of languages that are normally used to produce compiled programs include C, Fortran, and COBOL.

In an interpreted program, on the other hand, the source code typically is the program. Programs of this type (often known as scripts) require an interpreter, which parses the commands in the program and then executes them. Some interpreters, such as the Unix (agat) shells (agvf) (`sh` , `csh` , `ksh` , etc.), read and then immediately execute each command, while others, such as Perl, analyze the entire script before sending the corresponding machine language instructions. The advantage of a script is that it is very portable. Any computer that has the appropriate interpreter installed may run the program more or less unchanged. This is a disadvantage as well, because the program will not run at all if the interpreter is not available. In general, interpreted programs are slower than compiled programs, but are easier to debug and revise. Other examples of interpreted languages include JavaScript and Python (agvb).

Intermediate to computer-specific compiled programs and interpreted scripts are programs designed for runtime environments. Java and Smalltalk programs are executed in this fashion. Constructing programs for runtime environments is similar to writing traditional compiled programs. The difference is that instead of compiling the source code into a machine language, it is output into byte code for the runtime environment's "virtual machine". This virtual machine intercepts the byte code instructions and translates them into computer-specific commands. The advantage of this approach is that the runtime environment quickly compiles only the needed pieces of the code (some parts of the program may never need to be executed). This is called just-in-time compiling. The major disadvantage with runtime environments is that a program that is not designed well will force the runtime environment to compile almost all of the code up front and then make redundant calls to the interpreter. This makes the program slower to load and run.

Related documents

Compilers available on the IU research computing systems (abby)

This is document agsz *in the Knowledge Base.*
Last modified on 2018-01-18 12:17:56.

Contact us

For help or to comment, email the UITS Support Center (https://mailform.kb.iu.edu/email.php?cid=65).