Software portability

Portability in high-level computer programming is the usability of the same software in different environments. The prerequirement for portability is the generalized abstraction between the application logic and system interfaces. When software with the same functionality is produced for several computing platforms, portability is the key issue for development cost reduction.

Contents

Strategies for portability

Software portability may involve:

- Transferring installed program files to another computer of basically the same architecture.
- Reinstalling a program from distribution files on another computer of basically the same architecture.
- Building executable programs for different platforms from source code; this is what is usually understood by "porting".

Similar systems

When operating systems of the same family are installed on two computers with processors with similar instruction sets it is often possible to transfer the files implementing program files between them.

In the simplest case, the file or files may simply be copied from one machine to the other. However, in many cases, the software is installed on a computer in a way which depends upon its detailed hardware, software, and setup, with device drivers for particular devices, using installed operating system and supporting software components, and using different drives or directories.

In some cases, software, usually described as "portable software", is specifically designed to run on different computers with compatible operating systems and processors, without any machine-dependent installation. Porting is no more than transferring specified directories and their contents. Software installed on portable mass storage devices such as USB sticks can be used on any compatible computer on simply plugging the

storage device in, and stores all configuration information on the removable device. Hardware- and software-specific information is often stored in configuration files in specified locations (e.g. the registry on machines running Microsoft Windows).

Software which is not portable in this sense will have to be transferred with modifications to support the environment on the destination machine.

Different processors

As of 2011 the majority of desktop and laptop computers used microprocessors compatible with the 32- and 64-bit x86 instruction sets. Smaller portable devices use processors with different and incompatible instruction sets, such as ARM. The difference between larger and smaller devices is such that detailed software operation is different; an application designed to display suitably on a large screen cannot simply be ported to a pocket-sized smartphone with a tiny screen even if the functionality is similar.

Web applications are required to be processor independent, so portability can be achieved by using web programming techniques, writing in JavaScript. Such a program can run in a common web browser. Such web applications must, for security reasons, have limited control over the host computer, especially regarding reading and writing files. Non-web programs, installed upon a computer in the normal manner, can have more control, and yet achieve system portability by linking to portable libraries providing the same interface on different systems.

Source code portability

Software can be compiled and linked from source code for different operating systems and processors if written in a programming language supporting compilation for the platforms. This is usually a task for the program developers; typical users have neither access to the source code nor the required skills.

In open-source environments such as Linux the source code is available to all. In earlier days source code was often distributed in a standardised format, and could be built into executable code with a standard Make tool for any particular system by moderately knowledgeable users if no errors occurred during the build. Some Linux distributions distribute software to users in source form. In these cases there is usually no need for detailed adaptation of the software for the system; it is distributed in a way which modifies the compilation process to match the system.

Effort to port source code

Even with seemingly portable languages like C and C++ the effort to port source code can vary considerably. The authors of UNIX/32V (1979) reported that "[t]he (Bourne) shell [...] required by far the largest conversion effort of any supposedly portable program, for the simple reason that it is not portable."[1]

Sometimes the effort consists of recompiling the source code, but sometimes it is necessary to rewrite major parts of the software. Many language specifications describe implementation defined behaviour (e.g. right shifting a signed integer in C can do a logical or an arithmetic shift). Operating system functions or third party libraries might not be available on the target system. Some functions can be available on a target system, but exhibit slightly different behaviour (E.g.: utime() fails under Windows with EACCES, when it is called for a directory). The program code itself can also contain unportable things, like the paths of include files. Drive letters and the backslash as path delimiter are not accepted on all operating systems. Implementation defined things like byte order and the size of an int can also raise the porting effort. In practice the claim of languages, like C and C++, to have the WOCA (write once, compile anywhere) is arguable.

See also

- Interoperability
- Cross-platform software
- Hardware-dependent software
- C (programming language)
- Language interoperability
- Portability testing
- Source-to-source compiler
- Data portability

References

1. Thomas B. London and John F. Reiser (1978). A Unix operating system for the DEC VAX-11/780 computer (https://www.bell-labs.com/usr/dmr/www/otherports/32vscan.pdf). Bell Labs internal memo 78-1353-4.

Sources

- Mooney (1997). "Bringing Portability to the Software Process" (https://web.archive.org/web/20080725004932/http://www.cs.wvu.edu/~jdm/research/portability/reports/TR_97-1.pdf) (PDF). West Virginia University. Dept. of Statistics and Computer Science. Archived from the original (http://www.cs.wvu.edu/~jdm/research/portability/reports/TR_97-1.pdf) (PDF) on 2008-07-25. Retrieved 2008-03-17.
- Garen (2007). "Software Portability: Weighing Options, Making Choices" (http://findarticles.com/p/articles/mi_qa5346/is_200711/ai_n21298624/). *The CPA Journal*. **77** (11): 3.
- Lehey (1995). "Porting UNIX Software: From Download to Debug" (http://www.lemis.com/grog/Documentation/PUS/) (PDF). Retrieved 2010-05-27.