

Chapter 52

perlport

Writing portable Perl

52.1 DESCRIPTION

Perl runs on numerous operating systems. While most of them share much in common, they also have their own unique features.

This document is meant to help you to find out what constitutes portable Perl code. That way once you make a decision to write portably, you know where the lines are drawn, and you can stay within them.

There is a tradeoff between taking full advantage of one particular type of computer and taking advantage of a full range of them. Naturally, as you broaden your range and become more diverse, the common factors drop, and you are left with an increasingly smaller area of common ground in which you can operate to accomplish a particular task. Thus, when you begin attacking a problem, it is important to consider under which part of the tradeoff curve you want to operate. Specifically, you must decide whether it is important that the task that you are coding have the full generality of being portable, or whether to just get the job done right now. This is the hardest choice to be made. The rest is easy, because Perl provides many choices, whichever way you want to approach your problem.

Looking at it another way, writing portable code is usually about willfully limiting your available choices. Naturally, it takes discipline and sacrifice to do that. The product of portability and convenience may be a constant. You have been warned.

Be aware of two important points:

Not all Perl programs have to be portable

There is no reason you should not use Perl as a language to glue Unix tools together, or to prototype a Macintosh application, or to manage the Windows registry. If it makes no sense to aim for portability for one reason or another in a given program, then don't bother.

Nearly all of Perl already *is* portable

Don't be fooled into thinking that it is hard to create portable Perl code. It isn't. Perl tries its level-best to bridge the gaps between what's available on different platforms, and all the means available to use those features. Thus almost all Perl code runs on any machine without modification. But there are some significant issues in writing portable code, and this document is entirely about those issues.

Here's the general rule: When you approach a task commonly done using a whole range of platforms, think about writing portable code. That way, you don't sacrifice much by way of the implementation choices you can avail yourself of, and at the same time you can give your users lots of platform choices. On the other hand, when you have to take advantage of some unique feature of a particular platform, as is often the case with systems programming (whether for Unix, Windows, Mac OS, VMS, etc.), consider writing platform-specific code.

When the code will run on only two or three operating systems, you may need to consider only the differences of those particular systems. The important thing is to decide where the code will run and to be deliberate in your decision.