Editor: **Diomidis Spinellis**
Athens University of Economics and Business, dds@aueb.gr

Package Management Systems

Diomidis Spinellis

DYNAMIC LINKED LIBRARY (DLL) hell was a condition that often afflicted unfortunate users of old Microsoft Windows versions. Under it, the installation of one program would render others unusable due to incompatibilities between dynamically linked libraries. Suffering users would have to carefully juggle their conflicting DLLs to find a stable configuration. Similar problems distress any administrator manually installing software that depends on incompatible versions of other helper modules.

Thankfully, these problems are now ancient history for most of us thanks to the success of package management systems, which simplify software installation and maintenance by standardizing and organizing the production and consumption of software collections. Many modern operating systems and extensible applications use packages as the default software installation option; Table 1 lists commonly used ones.

A package management system comprises repositories hosting all packages, installation and maintenance software, and per-host data about installed packages. A package contains in a standardized format a software module's source or compiled code (or the place where these are hosted), together with its documentation and metadata. The metadata typically include the software's description, version, dependencies, vendor, license, and installation specifications. The system simplifies package installation by automatically fetching and installing not only the required package but also all its dependencies. Most package managers can also upgrade or remove installed packages. Maintaining packages in a centralized repository ensures that packages play well together and reduces the chance of version incompatibilities between dependencies. In fact, dependency checking has advanced to the point where some systems use a so-called Boolean satisfiability problem solver to verify that build dependencies aren't intractable.

Orderly and organized package management is a key element of a well-run software production process.

Consuming

As a software developer, you can benefit from package managers in two ways: through a rich and stable development environment and through friction-free reuse. Start with the development environment. By installing add-ons through a package manager, you can choose productivity-enhancing tools from an amazingly extensive selection. The stable distribution of Debian GNU/Linux offers 1,073 packaged tools that interest developers, starting from a56, a Motorola DSP 56001 assembler, to abi-compliance-checker, a tool to compare the application binary interface compatibility of shared C/C++ library versions, to zzuf, a transparent application fuzzer. On top of that, add a choice of 198 editors and development environments, 129 version management systems and utilities, 728 text-processing tools, and 25 shells. Or consider what CTAN, the comprehensive TeX archive, offers to documentation writers: if you want to add nicely formatted program listings, include multiple charts in a floating figure, or produce an index of cited authors, there's a package for that.

The success of packages as an installation mechanism has even led to the development of virtual packages or metaports—collections of actual packages often used together that can be installed in one simple step. For instance, the FreeBSD py27-kdebindings metaport will install 256 Python bindings to the KDE libraries, while the gnome2 metaport will install 481 packages comprising the GNOME 2 desktop and

0740-7459/12/$31.00 © 2012 IEEE

its most common user applications.

Installation through a package manager is fast and convenient. Although I used to enjoy porting and installing software by configuring and compiling its source code (porting Perl to MS-DOS was perversely interesting), I now have to be really desperate to install software in a form different from that of a managed package. With a package manager, you can easily document a development environment by simply listing the names of its installed packages. Through such a list, you can bring a new developer or development machine up to speed in less than an hour; gone are the detailed and ever-changing instructions on how to set up a working development environment. As icing on the cake, your environment is more likely to remain secure because most package managers provide facilities for upgrading all installed packages to their latest securely patched versions.

If you're going the package management installation route, you should also be aware of two potential problems. First, package managers create a closed ecosystem. If the software you want is not available as a package, it might be tricky to install it and make it play nicely with your other packages. Second, some package management systems trade life at the cutting edge for stability. This means that the software you install may well be a couple-years'-old release, which might not suit you if you're looking for the latest and greatest features.

TABLE 1

Commonly used package management systems

Name	Target platform	No. of packages
Debian packages	Debian GNU/Linux	35,900
Maven	Java	33,905
Portage	Gentoo Linux	27,000
FreeBSD Ports	FreeBSD Unix	22,900
CPAN	Perl	21,600
MacPorts	Mac OS X	13,300
Pkgsrc	15 operating systems	10,000
CRAN	R	4,300
NuGet	.NET	4,177
CTAN MiKTeX	TeX under Windows	2,300
Cygwin	Unix software under Windows	2,000

And Producing

Making your software work with a package manager isn't trivial and it isn't for everyone. Many package management systems work only with source code, some require the software to be licensed as open source, and none that I know of provide functionality allowing users to pay for installing a package. Therefore, if your software's business model isn't compatible with these constraints, you'll have to use an alternative distribution mechanism, perhaps through an app store, or enforce payment with a license manager. Then there are the hoops you have to jump through to get your software included in such a system. You'll have to provide all the needed metadata in the appropriate format, test that your software and its dependencies install and uninstall correctly on all supported systems, and convince the package maintainers to include your software in the repository. However, if you have a high-quality product that plays well with package management systems, you might actually find that a volunteer will create a package out of it.

There are many benefits in distributing your software as a package. First, you offer your users a higher-quality experience—all the goodies I described in the preceding section. Given a package's ease of installation, you can address many more and less sophisticated users. Many package management systems offer a GUI front end, and I suspect that for many users, this is the only way they install and manage their packages. Second, some packaging systems will download source code and compile it on the fly, and others offer precompiled binaries. Both approaches hide from you or your software's hapless users the complexity of configuring and running the compilation process. Thus, your software can easily run on diverse operating systems and processor architectures.

However, the biggest benefit of working within a package manager is the ability to reuse third-party components without guilt. I used to be wary and conservative about reusing libraries in my projects because they would introduce dependencies and maintenance responsibilities that created headaches both during software development and deployment. Within a package management ecosystem, installation of additional components becomes a nonissue. Ensuring ongoing compatibility between your code and a third-party component is also

IEEE Software

HOW TO REACH US

WRITERS

For detailed information on submitting articles, write for our Editorial Guidelines (software@computer.org) or access www.computer.org/software/author.htm.

LETTERS TO THE EDITOR

Send letters to

Editor, *IEEE Software*
10662 Los Vaqueros Circle
Los Alamitos, CA 90720
software@computer.org

Please provide an email address or daytime phone number with your letter.

ON THE WEB

www.computer.org/software

SUBSCRIBE

www.computer.org/software/subscribe

SUBSCRIPTION CHANGE OF ADDRESS

Send change-of-address requests for magazine subscriptions to address.change@ieee.org. Be sure to specify *IEEE Software.*

MEMBERSHIP CHANGE OF ADDRESS

Send change-of-address requests for IEEE and Computer Society membership to member.services@ieee.org.

MISSING OR DAMAGED COPIES

If you are missing an issue or you received a damaged copy, contact help@computer.org.

REPRINTS OF ARTICLES

For price information or to order reprints, send email to software@computer.org or fax +1 714 821 4010.

REPRINT PERMISSION

To obtain permission to reprint an article, contact the Intellectual Property Rights Office at copyrights@ieee.org.

less of a problem for two reasons: first, developers of widely reused packages are loathe to break backward compatibility because they'll face numerous complaints, and second, the package management system build infrastructure often detects these problems when an update is submitted and will ask the package maintainer to address them. And, yes, the detection of compatibility problems includes those associated with building your application, which is now part of the family.

The breadth of modules you can reuse is nothing short of amazing, making it a crime to start writing code before you investigate what packages you can reuse. Whatever your need, there's likely to be a package that you can effortlessly link with your application. Promisingly, the structure that package managers bring both to the tools we use in our development process and the libraries we reuse in our products ties nicely with the recent move emphasizing development operations (DevOps) as an integration between software development and IT operations. Orderly and organized package management is a key element of a well-run software production process. Maintaining a list of an organization's recommended packages allows teams to share best practices and avoids package incompatibilities. So, if your business allows it, join a package management ecosystem, enjoy the fruits of other people's labor, and contribute back to the community.

DIOMIDIS SPINELLIS is a professor in the Department of Management Science and Technology at the Athens University of Economics and Business and the author of the books *Code Reading* and *Code Quality: The Open Source Perspective* (Addison-Wesley, 2003, 2006). Contact him at dds@aueb.gr.

Post your comments online by visiting the column's blog:
www.spinellis.gr/tools

IEEE Software (ISSN 0740-7459) is published bimonthly by the IEEE Computer Society. IEEE headquarters: Three Park Ave., 17th Floor, New York, NY 10016-5997. IEEE Computer Society Publications Office: 10662 Los Vaqueros Cir., Los Alamitos, CA 90720-1314; +1 714 821 8380; fax +1 714 821 4010. IEEE Computer Society headquarters: 2001 L St., Ste. 700, Washington, DC 20036. Subscription rates: IEEE Computer Society members get the lowest rate of US$54 per year, which includes printed issues plus online access to all issues published since 1984. Go to www.computer.org/subscribe to order and for more information on other subscription prices. Back issues: $20 for members, $163 for nonmembers (plus shipping and handling).

Postmaster: Send undelivered copies and address changes to *IEEE Software*, Membership Processing Dept., IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08854-4141. Periodicals Postage Paid at New York, NY, and at additional mailing offices. Canadian GST #125634188. Canada Post Publications Mail Agreement Number 40013885. Return undeliverable Canadian addresses to PO Box 122, Niagara Falls, ON L2E 6S8, Canada. Printed in the USA.