《深入理解软件构造系统》

图书基本信息

书名：《深入理解软件构造系统》

13位ISBN编号：9787111382263

10位ISBN编号：7111382269

出版时间：2012-6-15

出版社：机械工业出版社华章公司

作者：Peter Smith

页数：406

译者：仲田

《深入理解软件构造系统》

内容概要

构造系统在软件开发过程中处于核心地位，它的正确性和性能，在一定程度上决定了软件开发成果的质量和软件开发过程的效率。本书作者作为一名软件构造系统专家，总结了自己在构造系统开发和维护方面的多年经验，对软件构造系统的原理进行了深入浅出的剖析，并通过各种实际使用场景，对几种最流行的构造工具进行了对比分析，另外还讨论了构造系统的性能优化、规模提升等高级主题。
本书分为四部分。第一部分：基础知识，第1~5章分别从构造系统的高层概念、基于Make的构造系统、程序的运行时视图、文件类型与编译工具、子标的与构造变量等方面介绍构造系统的概念和相关主题。第二部分：构造工具，第6~10章结合实际场景案例，对GNU Make、Ant、SCons、CMake和Eclipse IDE这五种构造工具进行分析比较，品评优劣，帮助读者了解构造工具的当前状况，并理解每种工具的优缺点。第三部分：高级主题，第11~16章对依赖关系、元数据、软件打包与安装、构造机器、工具管理等高级主题进行讨论，帮助读者理解关于建设构造系统的许多高级主题，并了解最佳实践。第四部分：提升规模，第17~19章讨论了在大规模构造系统的环境下，如何降低复杂性，提高构造运行速度，帮助读者理解如何设计出能够适应规模增长的小型构造系统，从而对软件构造系统有更好的认识。
本书适合软件开发相关人员，包含软件开发人员、项目经理、软件构造专业人士等阅读。

《深入理解软件构造系统》

作者简介

Peter Smith，资深软件开发工程师和软件构造系统专家，专注于软件生产效率的探索和研究，对各种新型软件工具的选用与开发、软件项目管理、IT基础设施项目管理、基于软件工具的流程改进，以及如何使企业的现有流程实现自动化等能帮助企业提高软件生产效率的一系列核心问题都有非常深入的认识，实践经验极为丰富。

Peter毕业于哥伦比亚大学，拥有计算机科学博士学位，研究方向是编译器和语言设计。他曾在大学任教，主要教授编译器设计、编程语言设计、软件工程和计算机网络等方面的课程。此外，他还是OOPSLA（面向对象编程、系统、语言与应用）协会的委员。

《深入理解软件构造系统》

书籍目录

《深入理解软件构造系统》

章节摘录

版权页：插图：14.2.2生成的文件被纳入到版本控制中如果目标文件或自动生成的源文件已经被提交到版本控制系统，这很可能是个错误。开发人员可能因为没看清楚，误将生成的文件当做源代码提交到版本控制系统中。如上文所述，如果生成的文件被不正确地与源代码保存在同一目录，开发人员就很容易犯这种错误。把生成的文件检入到版本控制系统，产生的一个副作用是：当第一个开发人员将这些文件提交后，所有其他开发人员都可能错误地提交这些文件。由于生成的文件是由构造系统自动写入的，当有人执行构造时，这些问题文件总会被修改。版本控制系统会注意到文件已修改，并准备再次将它提交到版本控制仓库中。如果开发人员不够细心，他们就会一次又一次地提交这些文件。为了尽快发现这一问题，可以考虑以只读模式检出文件（有些版本控制工具默认要求采用此模式）。当重新构造源树时，构造系统就无法写入这些生成的文件，从而导致构造失败。开发人员就会明白，这些文件是被误提交的，因此他们可以先从版本控制系统中删除这些文件，再继续进行构造。在某些情况下，把生成的文件提交到版本控制系统确实有意义，尽管这种想法一般是有问题的。例如，可以对构造树中不常变化的部分（例如第三方程序库）进行预编译，从而加速构造过程。通过预编译程序库并将结果提交到版本控制系统，就可以避免让每个开发人员都自行编译这些程序库。构造程序库必须使用特殊构造标的，从而使构造系统不会默认重新创建程序库，因此不会被标为“已修改”（进而不会被再次提交到版本控制系统）。某些版本控制工具对这种想法作了进一步扩展，它们可以自动缓存生成的文件，以节省开发人员的时间，不必重新构造这些文件。第19章将详细讨论这种机制。14.2.3构造管理脚本不应当纳入版本控制系统的最后一种场景，是当脚本或工具相对于产品源代码，更依赖于外部环境（例如构造机器或磁盘）时。把这种性质的工具提交到版本控制系统，会增加修正工具缺陷的维护工作量。例如，某脚本建议开发人员，当前何种构造机器是最快的，或何种文件系统有最大磁盘空间，那么该脚本不应提交到版本控制系统。软件的版本1和版本2没必要分别使用该脚本的不同版本。事实上，当修正该脚本的缺陷时，可能导致源代码的每个分支版本都必须修改。这当然是我们不希望看到的。因此，该脚本不应当纳入版本控制，而是应当保存为常规磁盘文件，例如/TOOLS/BIN/DISK-ADVISOR。对该脚本的任何修改（例如增加新的构造机器信息或新磁盘信息），可以单独进行。这同一个脚本可以用于所有代码分支，每个分支不需要分别使用不同的脚本。而且，该脚本只关注自身所处的当前构造环境，并不关心过去怎样。如果该脚本的某些行为依赖于代码的特定分支版本，那么还可以把该脚本所需的配置信息保存在版本控制系统，但把脚本主体仍放在/TOOLS/BIN目录。例如，如果DISK-ADVISOR脚本需要知道构造过程创建了哪些输出目录，则可以创建一个配置文件来列出这些信息。

《深入理解软件构造系统》

媒体关注与评论

《深入理解软件构造系统:原理与最佳实践》深入彻底地解析了软件的构造过程，包括软件构造过程中需要做出的各种选择、可能遇到的各种困难，以及原理与最佳实践。我不仅要向所有的软件构造工程师推荐这《深入理解软件构造系统:原理与最佳实践》，还要向所有的软件开发人员推荐，因为软件开发过程中有效性的关键在于具备一套精心设计的构造过程。——Kevin Bodie Pitney Bowes公司软件开发总裁《深入理解软件构造系统:原理与最佳实践》向我们清晰地展示了软件构造的原理与细节，内容涵盖构造软件产品需要用到的所有工具和技术，以及要避免的各种错误。无论是构造系统新手，还是经验丰富的构造系统工程师，《深入理解软件构造系统:原理与最佳实践》对他们来说都具有足够的吸引力。——Monte Davidoff Alluvial软件公司软件开发咨询师

《深入理解软件构造系统》

编辑推荐

《深入理解软件构造系统:原理与最佳实践》深入解析高效软件构造系统的实现原理和运作机制，及其可伸缩性和性能优化系统讲解实现和维护软件构造系统所需的理论、工具、流程、方法和技巧，以及各种常见的错误和陷阱，包含大量最佳实践。《深入理解软件构造系统:原理与最佳实践》深入解析了高效构造系统背后的核心原理，并调查研究了系统的特性和使用场景。《深入理解软件构造系统:原理与最佳实践》是作者多年创建并维护各种构造系统的经验结晶，能帮助我们在选择工具和方法时做出依据充分的决策，并避开常见的陷阱和错误。《深入理解软件构造系统:原理与最佳实践》还提供了丰富的实用示例，以及在JAVA、C、C++和C#等多种环境中总结的经验教训。《深入理解软件构造系统:原理与最佳实践》主要内容：系统讲解造系统的基础知识，包括源树、构造工具以及编译工具。比较5种领先的构造工具：GNUMake、Ant、SCons、CMake和EclipseIDE的集成构造特性。确保准确进行依赖关系检查，高效进行增量式构造。使用元数据帮助进行调试、性能分析和为源代码编制文档。打包软件，以备在目标机器上安装。包含管理复杂的版本控制系统、构造机器和编译工具的最佳实践。功能欠缺的构造系统可能会对开发人员的生产效率产生巨大的影响。错误的依赖关系、中断的编译错误、失效的软件实体、缓慢的编译速度，以及费时费力的手工处理，这些都是被人诟病的构造系统存在的问题。在本书中，软件生产效率专家Peter Smith向我们展示了如何实现能够解决以上所有问题的构造系统，使我们以更快的速度和更低的成本交付可靠的软件产品。如果你是一名开发人员，本书将向你展示在构造系统的建设和维护过程中涉及的各种问题，使之最符合团队的需要；如果你是一名管理人员，你会学习如何对团队的构造系统进行评估和效能改进；如果你是一名软件构造专家，无论面临多么严苛的要求，通过学习本书，你都能很好地优化构造系统的性能和可伸缩性。

《深入理解软件构造系统》

精彩短评

1、全面而理论的讲述了软件构造系统(build system)中的理论与常用的构建工具，Cpp 中的CMake,基础的Makefile等。
2、前端同学如果想了解build tool的水有多深可以看看这本，虽然由于作者的关系，广度还是有限…
3、介绍了软件系统设计过程中所需要用到的一系列工具
4、仔细读了一下，内容没有超出我理解的build工具，空洞无物
5、书还可以，但是有错误
6、很常用但又想不到会去专门研究的方面
7、花几个小时翻了一下，感觉很泛泛。很全很浅
8、贵！
9、以前很少关注底层的东东，同事推荐买的，粗略看了一遍，很有收获。基础还是要打牢靠了，才能往上盖高楼
10、开发一个烂的软件构造也许成本很低，但其维护的成本比开发的费用高的多。软件系统的构造很重要！

《深入理解软件构造系统》

版权说明