

深入理解软件构造系统



[深入理解软件构造系统_下载链接1](#)

著者:Peter Smith

出版者:机械工业出版社华章公司

出版时间:2012-6-15

装帧:平装

isbn:9787111382263

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。

作者介绍:

目录: 对本书的赞誉

译者序

前言

致谢

作者介绍

第一部分 基础知识

第1章 构造系统概述2

1.1 什么是构造系统2

1.1.1 编译型语言3

1.1.2 解释型语言3

1.1.3 Web应用4

1.1.4 单元测试5

1.1.5 静态分析5

1.1.6 文档生成6

1.2 构造系统的各个组成部分6

1.2.1 版本控制工具7

1.2.2 源树与目标树7

1.2.3 编译工具和构造工具8

1.2.4 构造机器9

1.2.5 发布打包与目标机器9

1.3 构造过程和构造描述11

1.4 如何使用构造系统12

构造管理工具12

1.5 构造系统的质量13

本章小结14

第2章 基于Make的构造系统15

2.1 Calculator示例15

2.2 创建一个简单的makefile17

2.3 对这个makefile进行简化19

2.4 额外的构造任务20

2.5 框架的运用21

本章小结23

第3章 程序的运行时视图24

3.1 可执行程序	24
3.1.1 原生机器码	25
3.1.2 单体系统镜像	25
3.1.3 程序完全解释执行	26
3.1.4 解释型字节码	26
3.2 程序库	28
3.2.1 静态链接	28
3.2.2 动态链接	29
3.3 配置文件和数据文件	30
3.4 分布式程序	30
本章小结	31
第4章 文件类型与编译工具	33
4.1 C/C++	34
4.1.1 编译工具	34
4.1.2 源文件	35
4.1.3 汇编语言文件	37
4.1.4 目标文件	38
4.1.5 可执行程序	40
4.1.6 静态程序库	40
4.1.7 动态程序库	41
4.1.8 C++编译	42
4.2 Java	43
4.2.1 编译工具	43
4.2.2 源文件	44
4.2.3 目标文件	45
4.2.4 可执行程序	47
4.2.5 程序库	48
4.3 C#	48
4.3.1 编译工具	49
4.3.2 源文件	49
4.3.3 可执行程序	51
4.3.4 程序库	53
4.4 其他文件类型	55
4.4.1 基于UML的代码生成	56
4.4.2 图形图像	57
4.4.3 XML配置文件	58
4.4.4 国际化与资源绑定	58
本章小结	59
第5章 子标的与构造变量	60
5.1 针对子标的进行构造	61
5.2 针对软件的不同版本进行构造	62
5.2.1 指定构造变量	63
5.2.2 对代码的定制调整	65
5.3 针对不同的目标系统架构进行构造	68
5.3.1 多重编译器	68
5.3.2 面向指定平台的文件／功能	69
5.3.3 多个目标树	69
本章小结	71
第二部分 构造工具	
现实场景	75
场景1：源代码放在单个目录中	75
场景2：源代码放在多个目录中	76
场景3：定义新的编译工具	76
场景4：针对多个变量进行构造	77

- 场景5：清除构造树77
- 场景6：对不正确的构造结果进行调试78
- 第6章 Make79
 - 6.1 GNU Make编程语言80
 - 6.1.1 makefile规则：用来建立依赖关系图80
 - 6.1.2 makefile规则的类型81
 - 6.1.3 makefile变量82
 - 6.1.4 内置变量和规则84
 - 6.1.5 数据结构与函数85
 - 6.1.6 理解程序流程87
 - 6.1.7 进一步阅读资料90
 - 6.2 现实世界的构造系统场景90
 - 6.2.1 场景1：源代码放在单个目录中90
 - 6.2.2 场景2(a)：源代码放在多个目录中92
 - 6.2.3 场景2(b)：对多个目录进行迭代式Make操作93
 - 6.2.4 场景2(c)：对多个目录进行包含式Make操作96
 - 6.2.5 场景3：定义新的编译工具101
 - 6.2.6 场景4：针对多个变量进行构造102
 - 6.2.7 场景5：清除构造树104
 - 6.2.8 场景6：对不正确的构造结果进行调试105
 - 6.3 赞扬与批评107
 - 6.3.1 赞扬107
 - 6.3.2 批评108
 - 6.3.3 评价109
 - 6.4 其他类似工具110
 - 6.4.1 Berkeley Make110
 - 6.4.2 NMake111
 - 6.4.3 ElectricAccelerator和Spark Build111
- 本章小结113
- 第7章 Ant115
 - 7.1 Ant编程语言116
 - 7.1.1 比“Hello World”稍多一些116
 - 7.1.2 标的的定义和使用118
 - 7.1.3 Ant的控制流119
 - 7.1.4 属性的定义120
 - 7.1.5 内置的和可选的任务122
 - 7.1.6 选择多个文件和目录125
 - 7.1.7 条件126
 - 7.1.8 扩展Ant语言127
 - 7.1.9 进一步阅读资料128
 - 7.2 现实世界的构造系统场景129
 - 7.2.1 场景1：源代码放在单个目录中129
 - 7.2.2 场景2(a)：源代码放在多个目录中130
 - 7.2.3 场景2(b)：多个目录，多个build.xml文件130
 - 7.2.4 场景3：定义新的编译工具133
 - 7.2.5 场景4：针对多个变量进行构造136
 - 7.2.6 场景5：清除构造树140
 - 7.2.7 场景6：对不正确的构造结果进行调试141
 - 7.3 赞扬与批评142
 - 7.3.1 赞扬143
 - 7.3.2 批评143
 - 7.3.3 评价144
 - 7.4 其他类似工具144
 - 7.4.1 NAnt144

- 7.4.2 MS Build145
- 本章小结146
- 第8章 SCons147
 - 8.1 SCons编程语言148
 - 8.1.1 Python编程语言148
 - 8.1.2 简单编译151
 - 8.1.3 管理构造环境154
 - 8.1.4 程序流程和依赖关系分析157
 - 8.1.5 决定何时重新编译158
 - 8.1.6 扩展该语言160
 - 8.1.7 其他有趣的特性162
 - 8.1.8 进一步阅读资料163
 - 8.2 现实世界的构造系统场景163
 - 8.2.1 场景1：源代码放在单个目录中163
 - 8.2.2 场景2(a)：源代码放在多个目录中163
 - 8.2.3 场景2(b)：多个SConstruct文件164
 - 8.2.4 场景3：定义新的编译工具165
 - 8.2.5 场景4：针对多个变量进行构造167
 - 8.2.6 场景5：清除构造树168
 - 8.2.7 场景6：对不正确的构造结果进行调试169
 - 8.3 赞扬与批评171
 - 8.3.1 赞扬171
 - 8.3.2 批评172
 - 8.3.3 评价173
 - 8.4 其他类似工具173
 - 8.4.1 Cons173
 - 8.4.2 Rake174
- 本章小结176
- 第9章 CMake177
 - 9.1 CMake编程语言178
 - 9.1.1 CMake语言基础178
 - 9.1.2 构造可执行程序 and 程序库179
 - 9.1.3 控制流182
 - 9.1.4 跨平台支持184
 - 9.1.5 生成原生构造系统185
 - 9.1.6 其他有趣的特性以及进一步阅读资料190
 - 9.2 现实世界的构造系统场景191
 - 9.2.1 场景1：源代码放在单个目录中191
 - 9.2.2 场景2：源代码放在多个目录中191
 - 9.2.3 场景3：定义新的编译工具192
 - 9.2.4 场景4：针对多个变量进行构造193
 - 9.2.5 场景5：清除构造树194
 - 9.2.6 场景6：对不正确的构造结果进行调试194
 - 9.3 赞扬与批评195
 - 9.3.1 赞扬195
 - 9.3.2 批评195
 - 9.3.3 评价196
 - 9.4 其他类似工具196
 - 9.4.1 Automake196
 - 9.4.2 Qmake197
- 本章小结197
- 第10章 Eclipse199
 - 10.1 Eclipse的概念和GUI199
 - 10.1.1 创建项目200

- 10.1.2 构造项目206
- 10.1.3 运行项目210
- 10.1.4 使用内部项目模型212
- 10.1.5 其他构造特性213
- 10.1.6 进一步阅读资料214
- 10.2 现实世界的构造系统场景215
 - 10.2.1 场景1：源代码放在单个目录中215
 - 10.2.2 场景2：源代码放在多个目录中216
 - 10.2.3 场景3：定义新的编译工具217
 - 10.2.4 场景4：针对多个变量进行构造217
 - 10.2.5 场景5：清除构造树220
 - 10.2.6 场景6：对不正确的构造结果进行调试220
- 10.3 赞扬与批评221
 - 10.3.1 赞扬221
 - 10.3.2 批评221
 - 10.3.3 评价222
- 10.4 其他类似工具222
- 本章小结224
- 第三部分 高级主题
- 第11章 依赖关系226
 - 11.1 依赖关系图227
 - 11.1.1 增量式编译228
 - 11.1.2 完全、增量式和子标的构造228
 - 11.2 依赖关系错误导致的问题229
 - 11.2.1 问题：依赖关系缺失导致运行时错误229
 - 11.2.2 问题：依赖关系缺失导致编译错误230
 - 11.2.3 问题：多余的依赖关系导致大量不必要的重新构造231
 - 11.2.4 问题：多余的依赖关系导致依赖关系分析失败231
 - 11.2.5 问题：循环依赖关系232
 - 11.2.6 问题：以隐式队列顺序替代依赖关系232
 - 11.2.7 问题：Clean标的什么也清除不了233
 - 11.3 步骤一：计算依赖关系图233
 - 11.3.1 获取确切的依赖关系234
 - 11.3.2 把依赖关系图缓存起来236
 - 11.3.3 对缓存的依赖关系图进行更新237
 - 11.4 步骤二：判断哪些文件已过期239
 - 11.4.1 基于时间戳的方法240
 - 11.4.2 基于校验和的方法241
 - 11.4.3 标志参数比较242
 - 11.4.4 其他高级方法243
 - 11.5 步骤三：为编译步骤排定队列顺序243
- 本章小结246
- 第12章 运用元数据进行构造247
 - 12.1 调试支持247
 - 12.2 性能分析支持249
 - 12.3 代码覆盖分析支持250
 - 12.4 源代码文档化251
 - 12.5 单元测试253
 - 12.6 静态分析256
 - 12.7 向构造系统加入元数据257
- 本章小结258
- 第13章 软件打包与安装259
 - 13.1 归档文件260
 - 13.1.1 用于打包的脚本260

- 13.1.2 其他归档文件格式262
- 13.1.3 对打包脚本的改进263
- 13.2 包管理工具265
 - 13.2.1 RPM包管理工具格式265
 - 13.2.2 rpm build过程266
 - 13.2.3 RPM规格文件示例267
 - 13.2.4 根据规格文件创建RPM文件272
 - 13.2.5 安装RPM示例274
- 13.3 定制式GUI安装工具275
 - 13.3.1 Nullsoft Scriptable Install System (NSIS) 276
 - 13.3.2 安装工具脚本277
 - 13.3.3 定义安装页面280
 - 13.3.4 许可授权页面281
 - 13.3.5 选择安装目录282
 - 13.3.6 主要组件282
 - 13.3.7 可选组件283
 - 13.3.8 定制页面285
 - 13.3.9 安装页面和卸载程序286
- 本章小结288
- 第14章 版本管理289
 - 14.1 对哪些东西进行版本控制290
 - 14.1.1 构造描述文件290
 - 14.1.2 对工具的引用292
 - 14.1.3 大型二进制文件296
 - 14.1.4 对源树的配置296
 - 14.2 哪些东西不应当放到源树中297
 - 14.2.1 生成的文件被保存到源树中297
 - 14.2.2 生成的文件被纳入到版本控制中299
 - 14.2.3 构造管理脚本299
 - 14.3 版本编号300
 - 14.3.1 版本编号体系300
 - 14.3.2 协调并更新版本号301
 - 14.3.3 版本号的保存与检索302
- 本章小结303
- 第15章 构造机器305
 - 15.1 原生编译与跨平台编译306
 - 15.1.1 原生编译306
 - 15.1.2 跨平台编译306
 - 15.1.3 异构环境307
 - 15.2 集中式开发环境307
 - 15.2.1 构造机器为何有差异308
 - 15.2.2 管理多个构造机器310
 - 15.3 开源开发环境312
 - 15.4 GNU Autoconf315
 - 15.4.1 高层次工作流315
 - 15.4.2 Autoconf示例317
 - 15.4.3 运行autoheader和autoconf319
 - 15.4.4 在构造机器上运行configure脚本320
 - 15.4.5 使用配置信息322
- 本章小结323
- 第16章 工具管理324
 - 16.1 工具管理的规则324
 - 16.1.1 规则1：做笔记324
 - 16.1.2 规则2：对源代码进行版本控制325

- 16.1.3 规则3：定期升级工具326
- 16.1.4 规则4：对工具的二进制文件进行版本控制327
- 16.1.5 对规则的破坏329
- 16.2 编写自己的编译工具329
- 用Lex和Yacc编写定制工具330
- 本章小结332
- 第四部分 提升规模
- 第17章 降低最终用户面对的复杂性334
 - 17.1 构造框架334
 - 17.1.1 面向开发人员的构造描述335
 - 17.1.2 面向框架的构造描述336
 - 17.1.3 惯例优先于配置336
 - 17.1.4 构造工具示例：Maven337
 - 17.2 避免支持多个构造变量的原因338
 - 17.2.1 需要测试更多的构造变量338
 - 17.2.2 代码会变得混乱339
 - 17.2.3 构造时间会增多340
 - 17.2.4 需要更多磁盘空间340
 - 17.3 降低复杂性的各种技术方法340
 - 17.3.1 使用现代构造工具340
 - 17.3.2 自动检测依赖关系341
 - 17.3.3 把生成的文件放在源树之外341
 - 17.3.4 确保正确清除构造树341
 - 17.3.5 碰到第一个错误即中止构造342
 - 17.3.6 提供有意义的错误信息343
 - 17.3.7 校验输入参数343
 - 17.3.8 不要把构造脚本搞得过分复杂344
 - 17.3.9 避免使用晦涩的语言特性344
 - 17.3.10 不要用环境变量控制构造过程345
 - 17.3.11 确保构造形成的发布版与调试版保持相似345
 - 17.3.12 准确显示正在执行的命令346
 - 17.3.13 把对工具的引用纳入版本控制347
 - 17.3.14 把构造指令纳入版本控制347
 - 17.3.15 自动检测编译标志参数的变化347
 - 17.3.16 不要在构造系统中调用版本控制工具347
 - 17.3.17 尽量频繁地进行持续集成348
 - 17.3.18 统一使用一种构造机器348
 - 17.3.19 统一使用一种编译器348
 - 17.3.20 避免遗留#ifdefs的垃圾代码348
 - 17.3.21 使用有意义的符号名349
 - 17.3.22 删除垃圾代码349
 - 17.3.23 不要复制源文件350
 - 17.3.24 使用统一的构造系统350
 - 17.4 对构造系统进行规划充分、人力充足的改进351
- 本章小结352
- 第18章 管理构造规模353
 - 18.1 单体模型构造存在的问题354
 - 18.2 组件式软件355
 - 18.2.1 使用组件的好处357
 - 18.2.2 组件到底是什么358
 - 18.2.3 把多个组件集成到单个产品中361
 - 18.3 人员和过程管理364
 - 18.3.1 开发团队的结构365
 - 18.3.2 组件版本队列管理367

18.3.3 管理组件缓存368
18.3.4 协调软件新特性的开发370
18.4 Apache Ivy372
本章小结373
第19章 更快的构造375
19.1 度量构造系统性能375
19.1.1 启动阶段的性能度量375
19.1.2 编译阶段的性能度量382
19.1.3 性能度量工具386
19.1.4 修正问题：改进性能388
19.2 构造减免：消除不必要的重新构造389
19.2.1 目标文件缓存389
19.2.2 智能依赖关系391
19.2.3 构造减免的其他技术方法395
19.3 并行396
19.3.1 构造集群／云396
19.3.2 并行构造工具397
19.3.3 对可伸缩性的限制398
19.4 减少磁盘使用398
本章小结400
参考文献401
• • • • • (收起)

[深入理解软件构造系统 下载链接1](#)

标签

软件工程

软件开发

构建

计算机科学

计算机

软件架构

编程

make

评论

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…

仔细读了一下，内容没有超出我理解的build工具，空洞无物

看来不错。

贵！

很常用但又想不到会去专门研究的方面

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。

我觉得这本书，写得浅显易懂。能够帮助大家，理解程序的构造过程。翻译的的也挺好的。

对我来说算是比较冷门的书了，但内容还是很详实的。对构造系统有一个全面的了解

- 深入理解软件构造系统 原理与最佳实践: 基础知识 构造工具(GNU make/Ant/SCons/CMake/Eclipse)

高级主题(依赖关系/元数据/软件打包/安装/构造机器/工具管理)
提升规模(降低复杂性/提高构造运行速度)

[深入理解软件构造系统 下载链接1](#)

书评

[深入理解软件构造系统 下载链接1](#)