



linux中apt-get使用

# apt-get简介

在Ubuntu系统中，经常要用到apt-get install指令来安装软件，由于常常需要root权限来操作，所以搭配sudo食用口感更佳，apt-get指令对于安装、卸载、升级软件提供一条龙服务，对比于源码安装，实在是业界良心。

## 源码安装

源码安装的流程一般是三部曲：

- ```
./configure
make
make install
```
- ./configure是为了检测目标安装平台的特征，并且检查依赖的软件包是否可用或者是否缺少依赖软件包，configure事实上是个脚本，最终的目的是生成Makefile。
  - 如果第一条指令没有报错，会生成一个Makefile，make指令就是编译这个源码包
  - 正常编译完之后如果没有报错，就生成了可执行文件，make install指令就是将可执行文件放到指定目录并配置环境变量，允许我们在任何目录下使用这个软件。

## 源码安装的优点

对于普通用户而言，实在是想不到什么优点...

对于软件开发者而言，可以拿到源码阅读学习并修改，geek一个软件简直比找女朋友还好玩！同时也可以在一定程度上防止黑客的攻击(你知道这个版本的漏洞，但是老夫已经把它修复了！！！)

## 源码安装的缺点

其实三部曲的安装还是不那么麻烦的，前提是不报错！一旦报错，对于linux初学者而言，那也真是丈二摸不着头脑，然后各种百度各种google，按照各种江湖术士的方法来整，结果把系统整崩的情况数不胜数，即使当时能用，但是也有可能留下在以后的使用中经常出现莫名其妙问题的隐患，让我们来看看这些问题都是啥样的：

- ./configure报错：如果检查到缺少依赖或者依赖文件的版本不匹配呢？一般出现这种情况，就自己解决吧，一般的做法是，升级软件包或者安装缺少的依赖软件包，运气好的话，解决报错的依赖问题就行了，运气不好的话，A软件包依赖B，B又依赖C.....这是比较常见的linux劝退方式，从入门到放弃！
- make报错，由于源码包的形式多是个人用户更新维护的，所以可能出现一些平台没测试到位或者在特定平台上程序出现bug的情况，这种情况就没办法了，如果你有能力debug那当然另说
- make install 报错，这个指令报错的形式一般仅仅是没有权限，加上sudo就行。但是同时因为源码包多由个人维护，也经常可能出现造成系统垃圾的情况，又或者你需要卸载的时候 make uninstall指令仅仅卸载可执行文件，其他配置文件和依赖文件不作处理。

## apt-get指令管理安装包

在上面说了那么多源码安装的缺点，聪明的盆友就要猜我我要引出今天的主角：apt-get包管理应用软件，由apt-get管理的软件包可以轻松做到一键安装卸载。废话不多说，我们先看看它的常用用法：

```
sudo apt-get install XXX
sudo apt-get install -y XXX
sudo apt-get install -q XXX
sudo apt-get remove XXX
sudo apt-get purge XXX
sudo apt-get autoremove
sudo apt-get update
sudo apt-get upgrade
```

## apt-get install

一键安装软件包，与源码安装不同的是，这个指令会自动检测并安装依赖，而且用apt-get安装的包都是成熟的软件包，基本不存在安装包有严重bug或者文件缺失的情况。

### 公告

昵称：牧野星辰  
园龄：1年7个月  
粉丝：31  
关注：2  
+加关注

| 2020年4月 |    |    |    |    |    |    |
|---------|----|----|----|----|----|----|
| <       | 日  | 一  | 二  | 三  | 四  | 五  |
| 29      | 30 | 31 | 1  | 2  | 3  | 4  |
| 5       | 6  | 7  | 8  | 9  | 10 | 11 |
| 12      | 13 | 14 | 15 | 16 | 17 | 18 |
| 19      | 20 | 21 | 22 | 23 | 24 | 25 |
| 26      | 27 | 28 | 29 | 30 | 1  | 2  |
| 3       | 4  | 5  | 6  | 7  | 8  | 9  |

### 搜索

找我看

谷歌搜索

### 常用链接

- 我的随笔
- 我的评论
- 我的参与
- 最新评论
- 我的标签

### 我的标签

- gcc 编译流程(2)
- github markdown(1)
- HTML标记(1)
- linux apt-get使用(1)
- linux bus(1)
- linux container\_of(1)
- linux i2c框架--i2c总线的初始化(1)
- linux i2c驱动程序源码实现(1)
- linux i2c驱动框架(1)
- linux nm工具的使用(1)
- 更多

### 随笔分类

- C(6)
- C++(5)
- linux设备驱动程序(17)
- Python(5)
- 操作系统(8)
- 生活
- 数据结构算法

sudo apt-get install -y

这里主要将的就是-y选项，添加这个选项就相当于不需要重复地确认安装

sudo apt-get install -q

即-quiet，静默安装，当然也不是完全静默，会将低等级的log信息屏蔽。

sudo apt-get remove

既然有安装就会有卸载，remove指令就是卸载，值得注意的是，remove仅仅卸载软件，但是并不卸载配置文件

sudo apt-get purge

卸载指令，同时卸载相应的配置文件

sudo apt-get autoremove

关于这条指令，官方解释是这样的：

```
autoremove is used to remove packages that were automatically installed to satisfy dependencies for other packages and are now no longer needed
```

在卸载软件的时候同时卸载那些当初作为依赖但是现在并不需要的包。

看起来非常完美的指令，但是博主**建议慎用**！！这条指令很可能将你要用的依赖包同时卸载，有时候你的安装包并没有通过apt-get指令来管理，apt-get管理工具不会加入这些包的信息，所以在检索包的依赖关系时可能出问题。

又或者是另一种情况：举个例子：在安装某个包时，这个包依赖git，但是git并非你主动下载的，而是作为依赖下载的，包安装完之后系统可能会提示git作为依赖不再需要使用，它并不知道你是不是正在使用这个软件包。

apt-get update

将所有包的来源更新，也就是提取最新的包信息，这一条我们经常使用到。

apt-get upgrade

这条指令一般执行在apt-get update之后，它的作用是将系统中旧版本的包升级成最新的，慎用！

因为在linux下，由于大部分为非商业软件，所以稳定性并没有得到很好的验证，升级到最新版本需要十分慎重！

apt-get执行原理

如果仅仅知道怎么用而不知道为什么这么用，那就违背了学习使用linux的初衷，所以我们还是需要从原理出发来看apt-get指令时怎么运行的。

提出问题

首先需要知道的问题是：

- 我们用apt-get下载的软件包是从哪里来的？
- 下载之前要做哪些准备工作？

软件从哪里来？

所有的deb包由官方统一管理，那为什么我们能定位到这些软件包呢？这里就涉及到一个软件源的概念，在/etc/apt/目录下有一个sources.list文件，我们来看一下这个文件的内容：

```
cat /etc/apt/sources.list

deb http://security.ubuntu.com/ubuntu trusty-security main restricted
deb-src http://security.ubuntu.com/ubuntu trusty-security main restricted
deb http://security.ubuntu.com/ubuntu trusty-security universe
deb-src http://security.ubuntu.com/ubuntu trusty-security universe
deb http://security.ubuntu.com/ubuntu trusty-security multiverse
deb-src http://security.ubuntu.com/ubuntu trusty-security multiverse
deb http://extras.ubuntu.com/ubuntu trusty main
deb-src http://extras.ubuntu.com/ubuntu trusty main
deb http://us.archive.ubuntu.com/ubuntu/ trusty-proposed main restricted multiverse universe
```

由于条目太多，这里只贴出一部分。可以看出的是，这里都是一些资源网站，软件包资源当然就是出自这里。

下载之前要做哪些工作

实践出真知，我们下载一个软件看看：

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
account-plugin-windows-live libblas3 libbonobo2-0 libbonobo2-common
```

数学(1)

随笔档案

- 2019年3月(40)
- 2018年9月(5)
- 2018年8月(1)

最新评论

- 1. Re:linux的initcall机制  
您好，我有一个疑问， for (fn = initcall\_levels[level]; fn < initcall\_levels[level+1]; fn++) do\_one\_initcall(\*f...  
--昔九
- 2. Re:linux的initcall机制  
写的很好，支持大佬！  
--昔九
- 3. Re:C++中string的实现原理  
注册一个账号来夸博主。有探索精神，向你学习。  
--张惊蛰
- 4. Re:浅谈操作系统与内存  
@ ismallboy游戏大小虽然超过4G如果运行内存不超过4g原则上应该可以运行...  
--人生尽得几何欢
- 5. Re:C++ STL hash表用法  
学习了，多谢  
--shiyideliutang

阅读排行榜

- 1. linux开机自启动(13611)
- 2. linux中apt-get使用(7235)
- 3. C++ STL hash表用法(7195)
- 4. linux内核模块编译makefile(4785)
- 5. python函数调用时参数传递方式(4741)

评论排行榜

- 1. linux的initcall机制(3)
- 2. linux设备驱动程序-设备树(2)-device\_no de转换成platform\_device(2)
- 3. 浅谈操作系统与内存(2)
- 4. 不同平台下int类型、指针类型的数据大小(2)
- 5. C++中string的实现原理(1)

推荐排行榜

- 1. linux的initcall机制(4)
- 2. linux内核makefile概览(2)
- 3. linux开机自启动(2)
- 4. C++中string的实现原理(2)
- 5. 浅谈操作系统与内存(2)

```
libbonoboui2-common libglade2-0 libgnomecanvas2-0 libgnomecanvas2-common
libgnomeui-common libidl-common libidl0 libisl10 liblinear-tools liblinear1
liblvm3.5 libntdb1 liborbit-2-0 liborbit2 libupstart1
linux-headers-3.16.0-30 linux-headers-3.16.0-30-generic
linux-image-3.16.0-30-generic linux-image-extra-3.16.0-30-generic
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
libdiodon0 libunique-3.0-0
Suggested packages:
diodon-plugins
The following NEW packages will be installed:
diodon libdiodon0 libunique-3.0-0
0 upgraded, 3 newly installed, 0 to remove and 221 not upgraded.
Need to get 272 kB of archives.
After this operation, 1,348 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

这是前期准备工作的log信息，我们可以看到

第一步是Reading package lists，这就是从/etc/apt/sources.list中检索可用的源中是否有这个软件包。

然后从下面的log可以看出，下面的步骤就是生成软件依赖树，将需要的依赖包提前列出来，在安装所需软件之前进行安装。

经常我们在下载软件的时候需要先添加源，很多盆友都不知道这是在干什么，看到这里大家应该是能看出个原因了：

因为你需要下载的软件并不在官方源网站中，所以需要先行添加资源网站，apt-get才能根据相应的源检索到相应的资源，添加源有很多操作方式，归根结底就是一个操作结果：在/etc/apt/sources.list添加相应的资源网站，知道了这个，就可以直接在文件中添加源，但是要记住linux下最基本的一个习惯：**操作系统文件时先备份**。

## 使用apt-get可能遇到的问题

在使用apt-get install的时候，我们可能会遇到下面的问题：

```
E: Could not get lock /var/lib/dpkg/lock - open (11: Resource temporarily unavailable)
E: Unable to lock the administration directory (/var/lib/dpkg/), is another process using it?
```

碰到这种问题怎么解决呢？

## 解决方法

这种问题的原因是apt-get被占用，无法再次使用apt-get命令操作。解决办法分为两种：

1、在终端输入，ps -ef | grep "apt-get",这个指令找出占用apt-get应用的进程，然后用sudo kill -9 PID强制结束进程

2、但是也有可能找不到这个占用的进程，很可能你在上一次安装操作的时候意外断电，没有正常退出导致的，这个时候需要做的就是强制删除以下文件就可以：

```
sudo rm /var/cache/apt/archives/lock
```

```
sudo rm /var/lib/dpkg/lock
```

从这里可以看出，其实就是删除两个锁文件，我们大可以猜想一下，系统在执行apt-get指令安装时，会在/var/lib/dpkg/和/var/cache/apt/archives/下生成lock文件，当指令正常退出时，系统会删除这个lock文件。

这样就可以解释为什么删除这两个文件就可以恢复apt-get的自由。

那我们就来验证一下，以安装diodon为例：

```
sudo apt-get install diodon
```

在系统询问是否继续时，我们先不操作，打开另一个终端。键入：

```
ls /var/lib/dpkg/lock
```

结果显示

```
/var/lib/dpkg/lock
```

表示是有这个文件的，然后我们关掉apt-get执行的进程，再检查没有操作apt-get文件时是否有这个文件：

```
ls /var/lib/dpkg/lock
```

结果显示

```
/var/lib/dpkg/lock
```

居然还能查到这个文件，证明我们的猜想是错误的！那就再猜测：既然是lock文件，那么就涉及到文件锁，文件是一直存在的，但是在执行apt-get时被锁住了，在正常退出的时候被解锁，立马行动，同样的，先键入：

```
sudo apt-get install diodon
```

并让其停在用户确认界面，在查看系统中被锁住的文件，在终端键入：

```
cat /proc/locks #查看系统中被锁定的文件
```

结果显示：

```
1: POSIX ADVISORY WRITE 13604 08:01:792131 0 EOF
2: POSIX ADVISORY WRITE 13604 08:01:792927 0 EOF
3: OFDLCK ADVISORY WRITE -1 08:01:393386 0 0
4: POSIX ADVISORY READ 2957 08:01:393504 128 128
5: POSIX ADVISORY READ 2957 08:01:393381 1073741826 1073742335
6: POSIX ADVISORY READ 2926 08:01:393504 128 128
7: POSIX ADVISORY READ 2926 08:01:393381 1073741826 1073742335
8: POSIX ADVISORY READ 2949 08:01:393504 128 128
9: POSIX ADVISORY READ 2949 08:01:393381 1073741826 1073742335
10: POSIX ADVISORY WRITE 2836 08:01:524297 0 EOF
```

...仅显示用户级别

然后关掉apt-get的进程，再次键入：

```
cat /proc/locks #查看系统中被锁定的文件
```

结果显示

```
1: OFDLCK ADVISORY WRITE -1 08:01:393386 0 0
2: POSIX ADVISORY READ 2957 08:01:393504 128 128
3: POSIX ADVISORY READ 2957 08:01:393381 1073741826 1073742335
4: POSIX ADVISORY READ 2926 08:01:393504 128 128
5: POSIX ADVISORY READ 2926 08:01:393381 1073741826 1073742335
6: POSIX ADVISORY READ 2949 08:01:393504 128 128
7: POSIX ADVISORY READ 2949 08:01:393381 1073741826 1073742335
8: POSIX ADVISORY WRITE 2836 08:01:524297 0 EOF
```

...#仅显示用户级别

对比发现很明显，在使用apt-get的时候，系统中被锁定的文件明显多出两个，但是那一行数字是什么意思呢？下列是对照关系，

| number, | type, | mode,    | type, | pid,  | maj:min:inode | start | end |
|---------|-------|----------|-------|-------|---------------|-------|-----|
| 1:      | POSIX | ADVISORY | WRITE | 13604 | 08:01:792131  | 0     | EOF |

我们可以通过inode的对比来确认在apt-get操作时多出来的两个被锁住的文件到底是不是上述提到的那两个锁文件。

```
ls -i /var/lib/dpkg/lock /var/cache/apt/archives/lock
```

输出结果：

```
792131 /var/cache/apt/archives/lock 792927 /var/lib/dpkg/lock
```

很显然，inode为792131,792927的两个文件恰好是上述进行apt-get操作和不操作时被加锁的两个文件，这个猜想是正确的。

但是问题又来了：当apt-get无法获取这两个锁文件的操作权限时，为什么删除这两个文件就可以了呢？这不会导致系统问题吗？

带着这个疑问，我们继续来操作：

```
sudo rm /var/cache/apt/archives/lock /var/lib/dpkg/lock
```

然后再查询文件：

```
ls /var/cache/apt/archives/lock /var/lib/dpkg/lock
```

输出结果：

```
ls: cannot access '/var/cache/apt/archives/lock': No such file or directory
ls: cannot access '/var/lib/dpkg/lock': No such file or directory
```

这下这两个文件是删除干净了，那我们得赶紧看看apt-get命令是否还能操作：

```
sudo apt-get install diodon
```

结果发现还是可以操作，这时候我再来看这两个文件：

```
ls /var/cache/apt/archives/lock /var/lib/dpkg/lock
```

输出结果：

```
/var/cache/apt/archives/lock /var/lib/dpkg/lock
```

居然又有了，这下是明白了，原来当我们遇到apt-get的无法获取锁的问题，直接删除这两个文件的同时这两个文件的锁自然也不存在了，在下一次重新使用apt-get指令时，系统检测到没有锁文件时就会创建这两个文件，同时进行apt-get操作。

其实这是一种覆盖式的偷懒操作，我们大可以用修改文件的锁属性的方式来解决这个问题，在文件inode结构体中有一个

```
struct file_lock *i_flock;
```

直接操作这个结构体就可以，但是这需要涉及到C和linux系统编程方面，在这里就不赘述了。

好了，关于ubuntu中apt-get包管理软件的讨论就到此为止啦，如果朋友们对于这个有什么疑问或者发现有文章中有什么错误，欢迎留言

原创博客，转载请注明出处！

祝各位早日实现项目丛中过，bug不沾身。

分类: [操作系统](#)

标签: [linux apt-get使用](#)

[好文要顶](#)[关注我](#)[收藏该文](#)

牧野星辰  
关注 - 2  
粉丝 - 31  
[+加关注](#)

1

推荐


0

反对

« 上一篇: [GCC编译流程浅析](#)  
» 下一篇: [linux开机自启动](#)

posted @ 2019-03-04 22:49 牧野星辰 阅读(7238) 评论(0) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)

 注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)， [访问](#) [网站首页](#)。

- 【推荐】超50万行VC++源码：大型组态工控、电力仿真CAD与GIS源码库
- 【推荐】10+实战课程看透面向对象的三大特性
- 【推荐】腾讯云产品限时秒杀，爆款1核2G云服务器99元/年！
- 【推荐】20本必看的阿里精品免费电子书



## 香港云服务器

免备案 · SSD存储 · 新一代CPU

29元起

# 1亿元红包

- 相关博文：
- [ubuntu apt-get用法](#)
  - [Linux apt-get](#)
  - [【Linux】-apt-get命令](#)
  - [Ubuntu\(Debian\)的aptitude与apt-get的区别和联系](#)
  - [Ubuntu之apt-get](#)
- » [更多推荐...](#)

[前端精品集合之JavaScript实战100例](#)



### cnblogs 专享，云服务器29元/月

广告 X

>

- 最新 IT 新闻：
- [微软 Bing 测试 Fluent Design 风格新 Logo](#)
  - [外媒：特斯拉上海工厂二期工程进展神速 似乎会比预期提前完工](#)
  - [比尔·盖茨：我们除了采取隔离别无选择，而且必须持续隔离一段时间](#)
  - [轻松筹：在噪音中狂奔](#)
  - [支付宝牵手全球最大公募基金 年轻人的第一次专业理财来了](#)
- » [更多新闻...](#)