**cmake\_tool使用手册**

**浪潮智能终端有限公司**

**2023.11**

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 修订日期 | 说明 | 修订人 |
| 0.1 | 2023-11-16 | <初始创建> | 赵洪宇 |
| 1.0 | 2023-11-23 | 修复bug，新增attach、detach、tar、untar子命令，部分子命令中新增-f、-a等功能选项 | 赵洪宇 |
|  |  |  |  |
|  |  |  |  |

**目录**

[1. 概述 1](#_Toc151651977)

[2. 安装cmake\_tool 2](#_Toc151651978)

[2.1. 源码编译安装（面向开发人员） 2](#_Toc151651979)

[2.2. 发布版安装（面向用户） 2](#_Toc151651980)

[2.3. 环境配置及启动 2](#_Toc151651981)

[3. 使用cmake\_tool 3](#_Toc151651982)

[3.1. 启动程序 3](#_Toc151651983)

[3.2. 创建cmake包 3](#_Toc151651984)

[3.3. 编译安装cmake包 4](#_Toc151651985)

[3.4. 清理cmake包 4](#_Toc151651986)

[3.5. 删除cmake包 5](#_Toc151651987)

[3.6. 运行cmake包的程序 5](#_Toc151651988)

[3.7. 列出cmake包 6](#_Toc151651989)

[3.8. 重置cmake\_tool的包信息 6](#_Toc151651990)

[3.9. 附加cmake\_tool的包信息 6](#_Toc151651991)

[3.10. 移除cmake\_tool的包信息 7](#_Toc151651992)

[3.11. 压缩cmake\_tool的包 8](#_Toc151651993)

[3.12. 解压tar包并附加到cmake\_tool 8](#_Toc151651994)

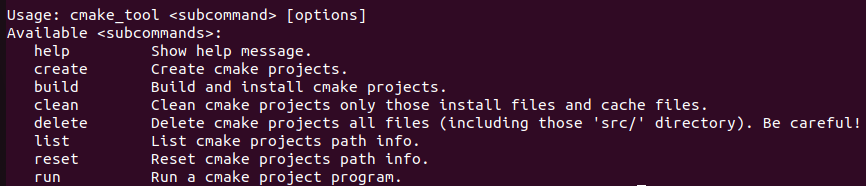
[4. 注意事项 9](#_Toc151651995)

[4.1. 同名包的处理 9](#_Toc151651996)

[4.2. 代码模板的定制 9](#_Toc151651997)

# 概述

cmake\_tool是一个cmake包的管理工具，可以新建cmake工程、编译安装cmake工程、清理cmake工程的编译安装、删除cmake工程、运行一个cmake工程的程序，等等。

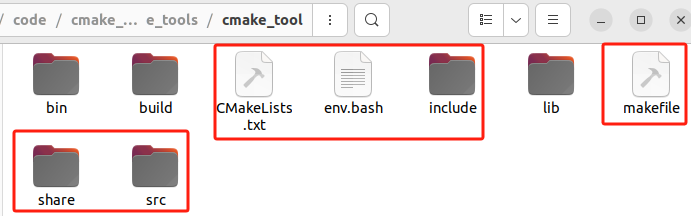


使用cmake\_tool可以避免用户从零开始搭建cmake工程，便于cmake工程的编译安装、运行和清理，同时，用户可以根据自己的需求场景，使用cmake\_tool定制自己cmake工程的模板。

# 安装cmake\_tool

## 源码编译安装（面向开发人员）

cmake\_tool源码的目录结构如下图所示：



被红色框框中的是编译cmake\_tool所需的源码，其中include/是存放头文件的目录，src/是存放cpp文件的目录，share/目录存放用户在使用cmake\_tool创建cmake工程时生成的代码模板（可修改定制）。

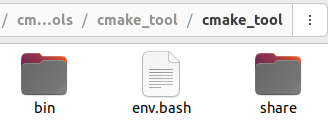
没被红色框框中的是编译后生成的目录， bin/目录存放cmake\_tool编译出的可执行文件，lib/目录存放cmake\_tool编译出的依赖库。

cmake\_tool的编译安装依靠源码目录的makefile脚本，步骤如下：

1. 使用cd命令进入到cmake\_tool的源码目录
2. 执行make命令，会自动编译cmake\_tool并将可执行程序安装到源码目录的bin/文件夹中

## 发布版安装（面向用户）

cmake\_tool以压缩包的方式发布，直接解压压缩包即完成安装，解压后的目录结构如下图所示：



bin/目录存放了可执行文件cmake\_tool，share/目录存放了用户在使用cmake\_tool创建cmake工程时生成的代码模板（可修改定制），env.bash用于配置cmake\_tool的环境变量以及双击tab键补全的功能。

## 环境配置及启动

得到cmake\_tool的可执行文件后，即可启动cmake\_tool，启动方式有以下两种：

方式1：直接执行cmake\_tool/bin目录中的cmake\_tool可执行程序：

1. /.../cmake\_tool/bin/cmake\_tool

方式2：先source cmake\_tool源码目录中的env.bash，（或者直接将这个source语句添加到~/.bashrc中），配置cmake\_tool的环境变量，然后在任意目录下直接输入cmake\_tool即可启动可执行程序：

1. source /.../cmake\_tool/env.bash
2. cmake\_tool

推荐使用方式2，先配置cmake\_tool的环境变量，再运行cmake\_tool，因为cmake\_tool在配置环境变量以后可以使用双击tab键进行命令补全的功能。

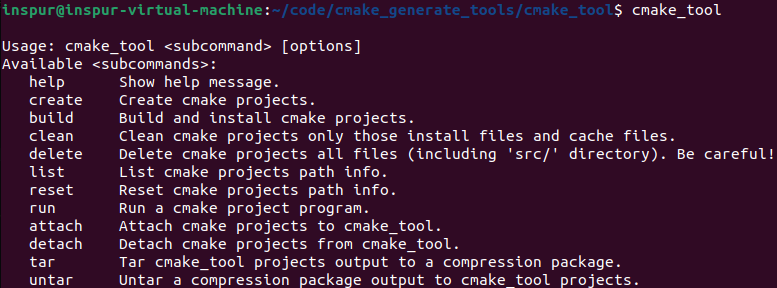
# 使用cmake\_tool

## 启动程序

**功能简介：**启动cmake\_tool，后面不带子命令时会显示帮助信息，加上子命令时执行对应的子命令，命令输入有误时也会显示帮助信息。如果source了 env.bash环境变量，cmake\_tool可以双击tab键进行命令补全。

**命令格式：**cmake\_tool <subcommand> [options]

**使用说明：**

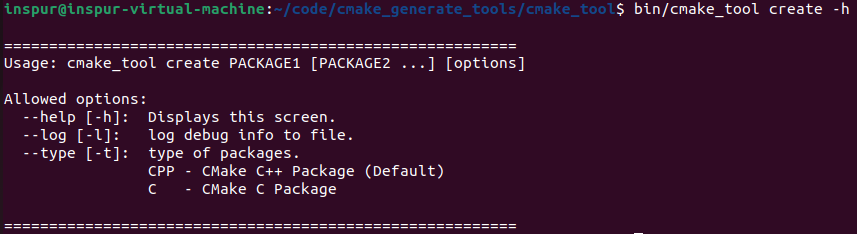


## 创建cmake包

**功能简介：**创建一个或多个cmake包。

**命令格式：**cmake\_tool create PACKAGE1 [PACKAGE2…] [options]

**使用说明：**



**使用示例：**

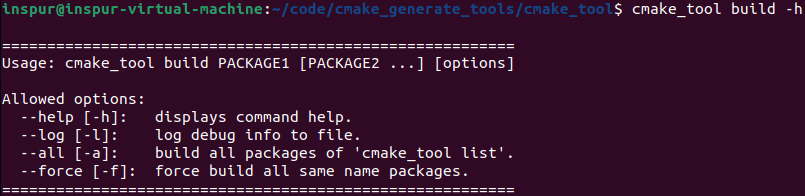
1. cmake\_tool create –h // 显示create的使用说明
2. cmake\_tool create test1 –l // 在当前目录创建test1包，并记录日志文件
3. cmake\_tool create test1 ~/test2 –t C // 创建./test1和~/test2包，包的类型为C语言（默认是C++）

## 编译安装cmake包

**功能简介：**编译并安装一个或多个cmake包。

**命令格式：**cmake\_tool build PACKAGE1 [PACKAGE2…] [options]

**使用说明：**



**使用示例：**

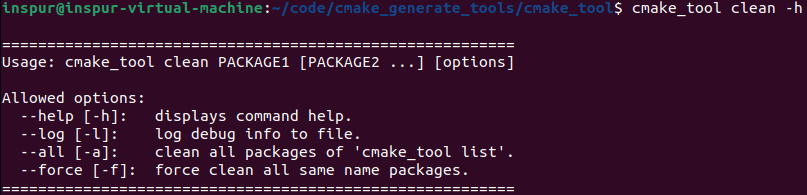
1. cmake\_tool build test1 ~/test2 // 依次编译和安装./test1、~/test2包
2. cmake\_tool build –a // 依次编译和安装cmake\_tool管理的所有包
3. cmake\_tool build test1 –f // 强制编译和安装cmake\_tool管理的所有包名为test1的包

## 清理cmake包

**功能简介：**清理一个或多个cmake包的编译安装文件，恢复到build之前的状态。

**命令格式：**cmake\_tool clean PACKAGE1 [PACKAGE2…] [options]

**使用说明：**



**使用示例：**

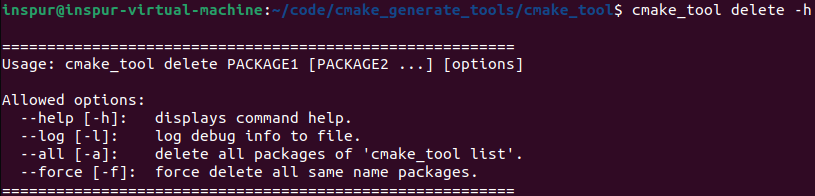
1. cmake\_tool clean test1 ~/test2 // 依次清理./test1、~/test2包的编译安装文件
2. cmake\_tool clean –a // 依次清理cmake\_tool管理的所有包
3. cmake\_tool clean test1 –f // 强制清理cmake\_tool管理的所有包名为test1的包

## 删除cmake包

**功能简介：**删除一个或多个cmake包。

**命令格式：**cmake\_tool delete PACKAGE1 [PACKAGE2…] [options]

**使用说明：**



**使用示例：**

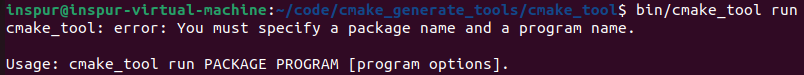
1. cmake\_tool delete test1 ~/test2 // 依次删除./test1、~/test2包的所有内容（包括源码），删除时会有提示，输入y后才会删除
2. cmake\_tool delete –a // 依次删除cmake\_tool管理的所有包
3. cmake\_tool delete test1 –f // 强制删除cmake\_tool管理的所有包名为test1的包

## 运行cmake包的程序

**功能简介：**运行一个cmake包中的可执行程序。

**命令格式：**cmake\_tool run PACKAGE PROGRAM [program options]

**使用说明：**



**使用示例：**

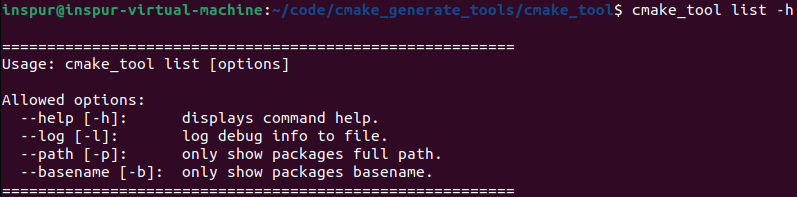
1. cmake\_tool run test1 program1 // 运行./test1包中编译出的可执行程序program1，如果program1运行时需要输入参数，将这些参数直接追加在命令后面

## 列出cmake包

**功能简介：**列出当前机器中cmake\_tool管理的所有包的信息。

**命令格式：**cmake\_tool list [options]

**使用说明：**



**使用示例：**

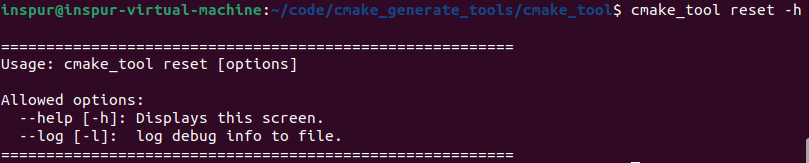
1. cmake\_tool list // 列出当前机器中cmake\_tool管理的所有包名及路径
2. cmake\_tool list –p // 只列出当前机器中cmake\_tool管理的所有包的全路径
3. cmake\_tool list –b // 只列出当前机器中cmake\_tool管理的所有包的包名

## 重置cmake\_tool的包信息

**功能简介：**当发现cmake\_tool中的包信息与本地机器不一致时（比如cmake\_tool list列出的包，在本机器上并不存在），应使用reset命令对cmake\_tool管理的包信息进行重置。

**命令格式：**cmake\_tool reset [options]

**使用说明：**



**使用示例：**

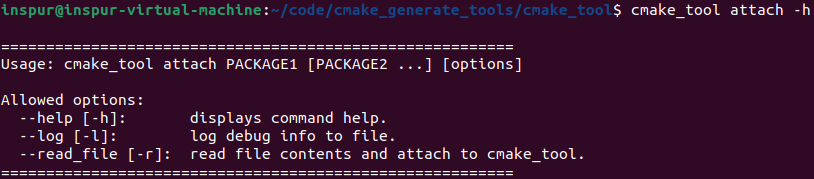
1. cmake\_tool reset // 重置cmake\_tool管理的包信息

## 附加cmake\_tool的包信息

**功能简介：**将本地机器的一个或多个cmake包，附加到cmake\_tool的包信息中，附加之后便可以通过cmake\_tool工具管理和运行这些包。注意：附加的cmake包应满足cmake\_tool包的格式，即可执行程序位于bin/目录，依赖库位于lib/目录。

**命令格式：**cmake\_tool attach PACKAGE1 [PACKAGE2…] [options]

**使用说明：**



**使用示例：**

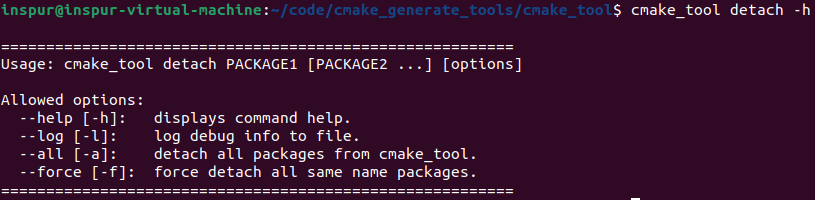
1. cmake\_tool attach test1 ~/test2 // 将./test1和~/test2附加到cmake\_tool中管理
2. cmake\_tool attach test1 –r packages.txt // 将./test1和./packages.txt文件中记录的包附加到cmake\_tool中管理

## 移除cmake\_tool的包信息

**功能简介：**将一个或多个cmake包从cmake\_tool的包信息中移除，移除之后这些包将脱离cmake\_tool工具的控制。注意：移除的cmake包如果在本地机器中存在，会提示用户确认是否移除，输入y或yes后才会移除。

**命令格式：**cmake\_tool detach PACKAGE1 [PACKAGE2…] [options]

**使用说明：**



**使用示例：**

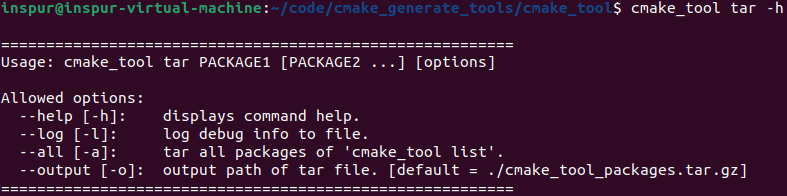
1. cmake\_tool detach test1 ~/test2 // 将./test1和~/test2脱离cmake\_tool的控制
2. cmake\_tool detach test1 –f // 强制将所有包名为test1的包脱离cmake\_tool的控制
3. cmake\_tool detach –a // 将所有包脱离cmake\_tool的控制，执行该命令后会在share/目录创建一个备份文件create.info.bak，可以用于恢复cmake\_tool的包信息

## 压缩cmake\_tool的包

**功能简介：**将本地机器的一个或多个cmake包，压缩为tar包，默认压缩为./cmake\_tool\_packages.tar.gz，可以使用-o选项指定tar包输出的路径名，路径名可以是tar包输出的目录，也可以是tar包输出的全路径（包含tar包的名字）。

**命令格式：**cmake\_tool tar PACKAGE1 [PACKAGE2…] [options]

**使用说明：**



**使用示例：**

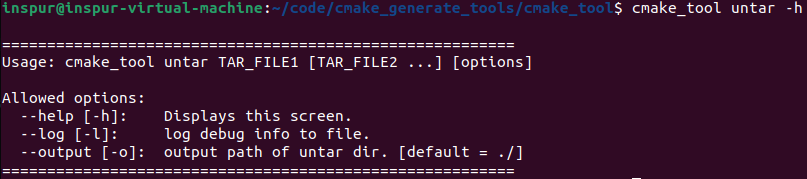
1. cmake\_tool tar test1 ~/test2 // 将./test1和~/test2压缩为./cmake\_tool\_packages.tar.gz
2. cmake\_tool tar test1 ~/test2 –o ~ // 将./test1和~/test2压缩为~/cmake\_tool\_packages.tar.gz
3. cmake\_tool tar test1 ~/test2 –o ~/test.tar.gz // 将./test1和~/test2压缩为~/test.tar.gz
4. cmake\_tool tar –a –o ~/test.tar.gz // 将cmake\_tool管理的所有包压缩为~/test.tar.gz

## 解压tar包并附加到cmake\_tool

**功能简介：**将本地机器的一个或多个tar压缩包，解压到输出目录，并将tar包中的cmake包附加到cmake\_tool，输出目录默认为./，可以使用-o选项指定输出目录。注意：tar压缩包中的内容应包含一个或多个满足cmake\_tool包格式的cmake包。

**命令格式：**cmake\_tool untar TAR\_FILE1 [TAR\_FILE2…] [options]

**使用说明：**



**使用示例：**

1. cmake\_tool untar cmake\_tool\_packages.tar.gz –o ~ // 将./cmake\_tool\_packages.tar.gz解压到~/，并将解压得到的包附加到cmake\_tool
2. cmake\_tool untar ~/test.tar.gz // 将~/test.tar.gz解压到./，并将解压得到的包附加到cmake\_tool

# 注意事项

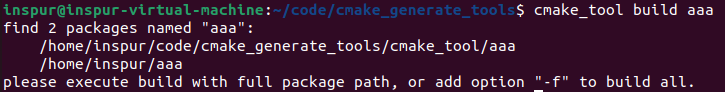
## 同名包的处理

在一台机器中，可能存在两个名字相同的cmake包，分别位于两个不同的目录，这种情况要指定某个包进行操作，为了防止出现歧义，最好使用绝对路径指定包。



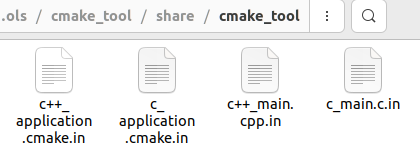
如果没有使用绝对路径，仅使用包名，默认会操作当前目录中的包。

如果当前目录中不存在该包，将不会对同名包进行任何操作，并提示用户使用绝对路径重新运行命令，也可以使用-f选项强制处理所有同名包。



## 代码模板的定制

前文中说过，在cmake\_tool源码目录中有一个share文件夹，文件夹里存放了每种类型对应的模板文件，目前只有C++和C两种类型，每种类型的模板文件有两个，分别是后缀名为.cmake.in的文件（用于定制cmake包的CMakeLists.txt），后缀名为.cpp.in的文件（用于定制cmake包的main函数所在的cpp文件）。



用户可以根据自己的需求场景，更改模板文件的内容，此后使用cmake\_tool创建的cmake包的内容（CMakeLists.txt和.cpp文件）将会更改为新的模板内容。