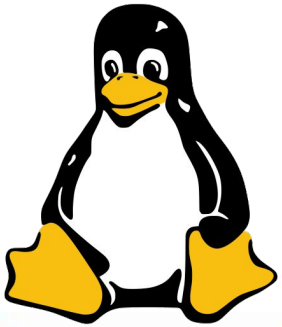
重温
debian打包

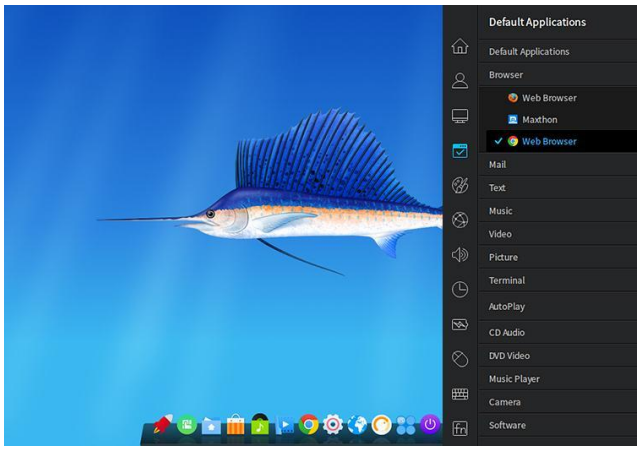
子影
Deepin
yhsponder@gmail.com

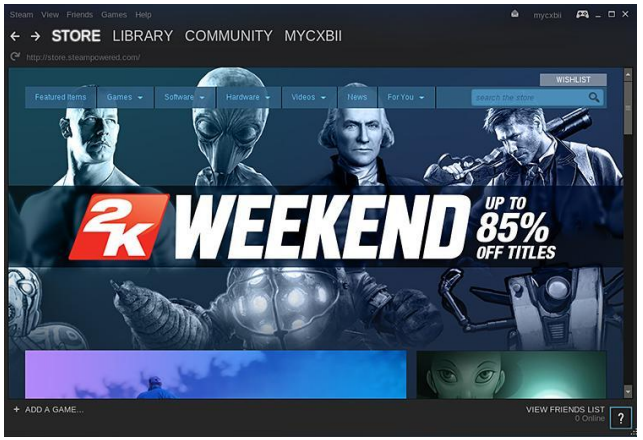


deepin

deepin

- 国内最活跃的Linux操作系统开发厂商
- 提供开箱即用的桌面操作系统体验
- 完善linux的应用生态
- 全球排名前10的活跃Linux发行版本





知识大纲

温习打包基础知识

- dh-make
- Changelog
- Control
- Rules
- Other
- Dpkg-buildpackage

什么是Debian包

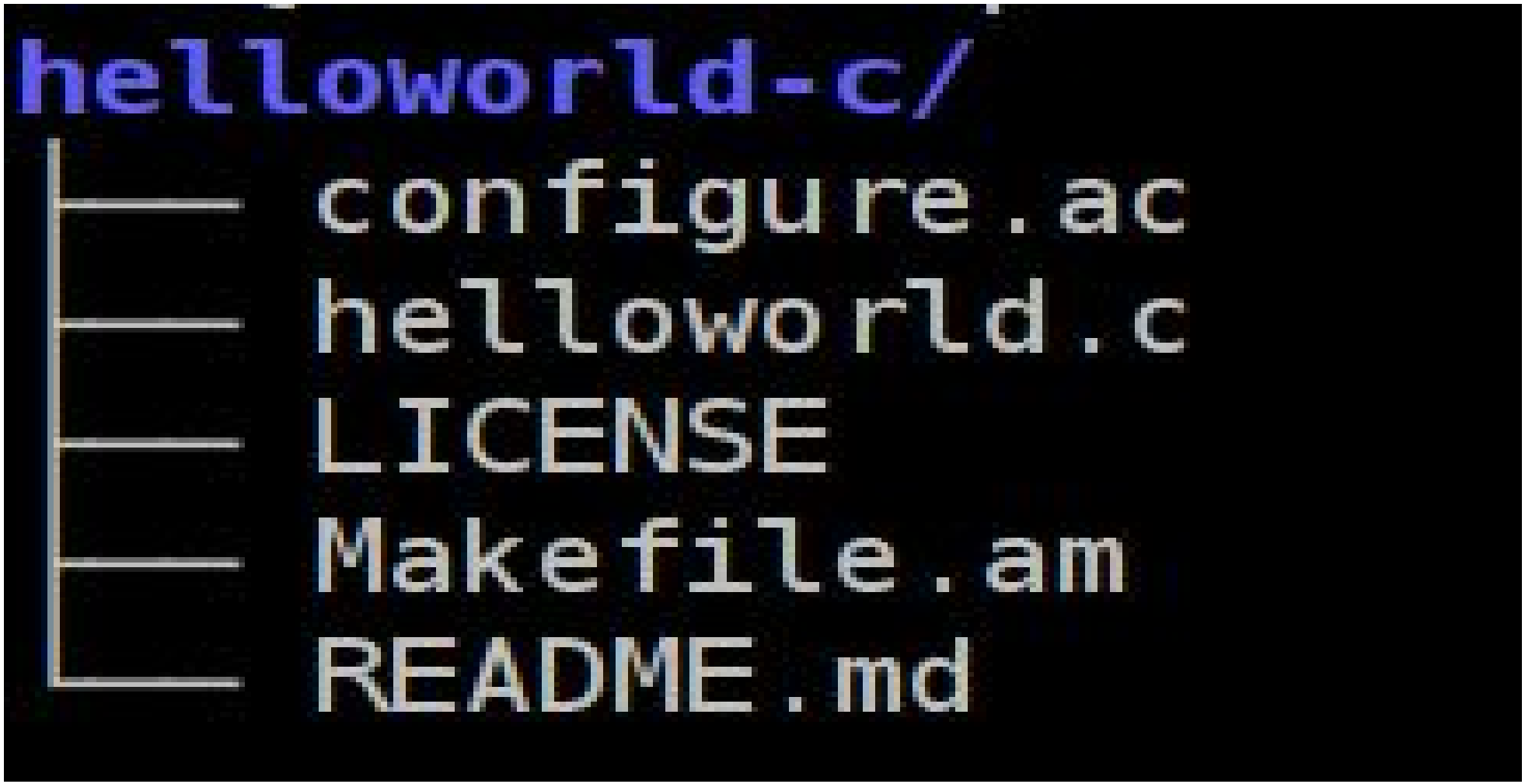
```
root@deepin# ar tvf helloworld_0.1_amd64.deb
rw-r--r-- 0/0        4 Jul 26 16:10 2017 debian-binary
rw-r--r-- 0/0      551 Jul 26 16:10 2017 control.tar.gz
rw-r--r-- 0/0     2728 Jul 26 16:10 2017 data.tar.xz
```

第一个栗子

helloworld

```
#include <stdio.h>
int main(void)
{
   printf("Hello, world!\n");
   return 0;
}
```

第一个栗子


helloworld-c/
configure.ac
helloworld.c
LICENSE
Makefile.am
README.md

第一个栗子

git clone
https://github.com/heysion/helloworld-c.git

aclocal
autoconf
automake --add-missing
./configure
make dist

dh-make

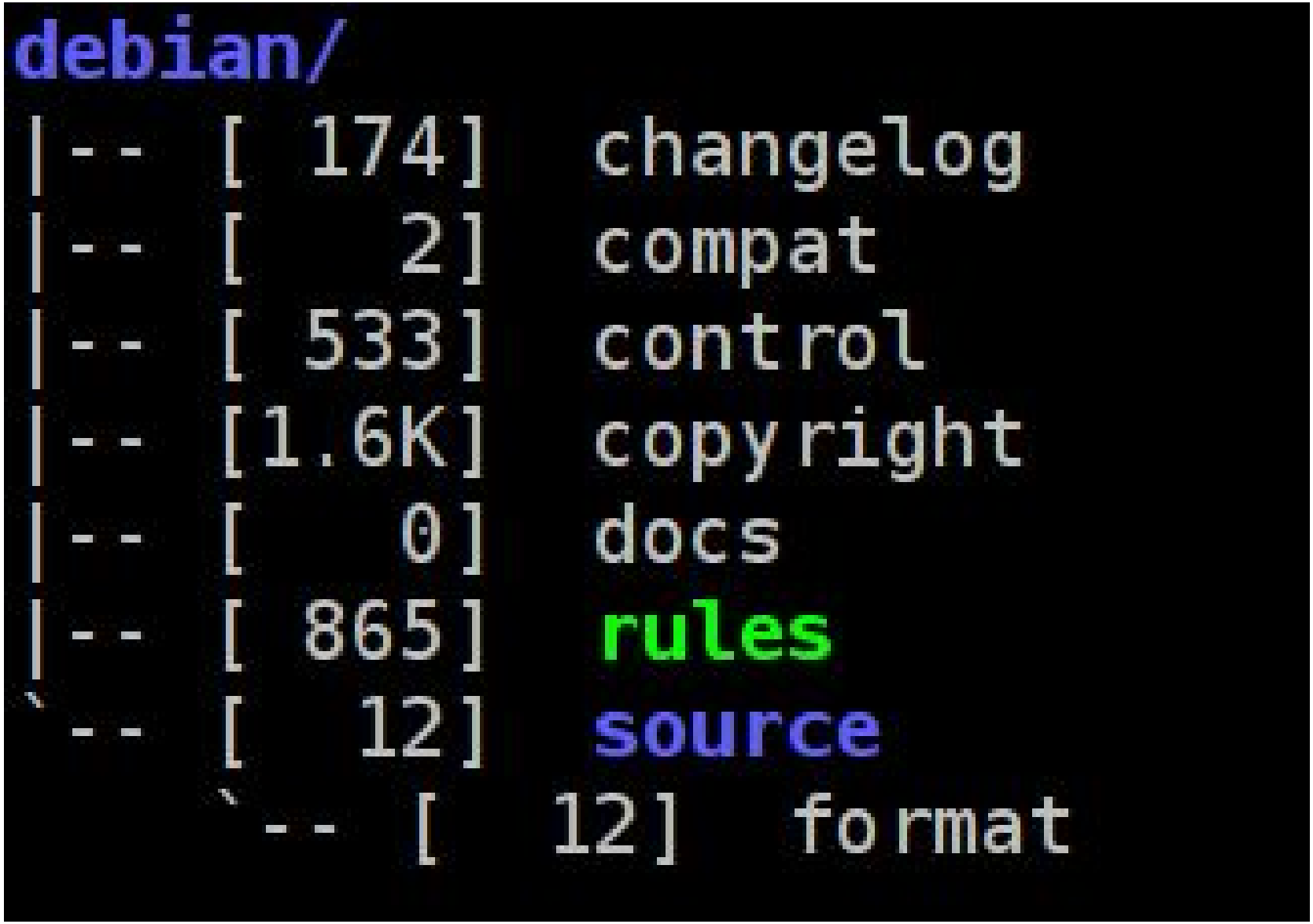
debian打包预处理工具

dh_make -f ../helloworld-0.1.tar.gz

dh-make

- single binary 单包 常用选项
- indep binary 架构无关的包 all包
- multiple binary 多包选项
- library 库包
- kernel module 内核模块包
- kernel patch 内核补丁包

debian目录


debian/
|-- [174] changelog
|-- [2] compat
|-- [533] control
|-- [1.6K] copyright
|-- [0] docs
|-- [865] rules
`-- [12] source
`-- [12] format

Changelog

用于描述上游的软件包的变更情况.

- 摘要.
- 修复bug的情况
- 新增的特性
- 移除的特性
- 等等

Changelog

```
helloworld (0.1-1) unstable; urgency=low

  * Initial release (Closes: #nnnn)  <nnnn is the bug number of your ITP>

 -- root <root@unknown>  Sat, 22 Jul 2017 19:11:07 +0000
debian/changelog (END)
```

Changelog

- package包名字
- version 版本号
- distribution 发行版本代号
- urgency 紧急程度
- 变更内容
- 维护人员信息

Compat

debian打包套件工具的兼容版本号

Copyright

描述软件的许可版本信息.

- 软件名字
- 维护者
- 源码仓库
- 公开授权类型
- 等.

Source/format

源包的版本格式

目前常用两种

- 3.0 (native) - Debian 与 源码在同一个压缩包里面
- 3.0 (quilt) - 源码和debian分开.patch 与上游源码分开.

Control

描述包的各种主要信息

- 控制包的原始信息
- 包分类
- 编译依赖
- 运行依赖
- 等等

Control

```
Source: helloworld
Section: unknown
Priority: optional
Maintainer: root <root@unknown>
Build-Depends: debhelper (>= 9), autotools-dev
Standards-Version: 3.9.5
Homepage: <insert the upstream URL, if relevant>
#Vcs-Git: git://anonscm.debian.org/collab-maint/helloworld.git
#Vcs-Browser: http://anonscm.debian.org/?p=collab-maint/helloworld.git;a=summary

Package: helloworld
Architecture: any
Depends: ${shlibs:Depends}, ${misc:Depends}
Description: <insert up to 60 chars description>
 <insert long description, indented with spaces>
debian/control (END)
```

Control

Source : 源码包名字

Section : 包的分类

Priority : 包的优先级

Maintainer : 包的维护者信息

Standards-version : 打包policy文档标准版本

Homepage : 软件的相关页面

Description : 软件信息描述

Control

Build-Depends ：编译依赖

软件包版本限定符

<<、<=、=、>= 和 >>

a (> 1.0) , b , c

软件包关系

a | b , c

Control

Architecture : 目标架构

- any 与编译环境建构相同
- all 全平台
- 限定架构 amd64 i386

Control

Depends : 运行依赖

- shlibs:depends
- misc:depends

Control

包关系控制

- Pre-Depends：前置依赖
- Recommends：推荐安装的软件依赖
- Suggests：推荐依赖软件
- Breaks：被覆盖的软件包
- Conflicts：与冲突的软件包
- Provides：提供某个包的功能
- Replaces：替换掉某个包的存在

Rules

- 描述debian包构建规则
- 与Makefile类似。
- debian打包里面最灵活的东西
- 打包最复杂的东西之一

Rules

debian/rules

```
grep -v ^# debian/rules  | sed -e '/^$/d'
DPKG_EXPORT_BUILDFLAGS = 1
include /usr/share/dpkg/default.mk
%:
    dh $@  --with autotools-dev
```

Rules

第一个过程：dh工具初始化过程

- dpkg-buildpackage: source package helloworld
- dpkg-buildpackage: source version 0.1
- dpkg-buildpackage: source distribution unstable
- dpkg-buildpackage: source changed by root <root@unknown>
- dpkg-buildpackage: host architecture amd64
- dpkg-source --before-build helloworld-0.1

Rules

第二个过程：dh clean过程

debian/rules clean

- dh clean --with autotools-dev
  - dh_testdir
  - dh_auto_clean
  - dh_autotools-dev_restoreconfig
  - dh_clean

Rules

重定义dh_auto_clean动作

```
override_dh_auto_clean:
        echo “redefine dh auto clean”
        find . -name “*.o” -exec rm -rfv {} \;
override_xxx
```

Rules

提交修改源码

dpkg-source --commit

Rules

第三个过程：dh build过程

debian/rules build

- dh build --with autotools-dev
  - dh_testdir
  - dh_autotools-dev_updateconfig
  - dh_auto_configure
  - dh_auto_build
  - dh_auto_test

Rules

自定义configure动作

```
override_dh_auto_configure:
	dh_auto_configure -- --prefix=/usr/local/
```

Rules

第四个过程：dh binary过程

1. dh_testroot
2. dh_prep
3. dh_auto_install
4. dh_installdocs
5. dh_installchangelogs
6. dh_perl
7. dh_link

Rules

第四个过程：dh binary过程

1. dh_testroot
2. dh_prep
3. dh_auto_install
4. dh_installdocs
5. dh_installchangelogs
6. dh_perl
7. dh_link

Rules

第四个过程：dh binary过程

8. dh_compress
9. dh_fixperms
10. dh_strip
11. dh_makeshlibs
12. dh_shlibdeps
13. dh_installdeb

Rules

第四个过程：dh binary过程

14. dh_gencontrol
15. dh_md5sums
16. dh_builddeb

Rules

第五个过程：收尾阶段

dpkg-genchanges >../helloworld_0.1_amd64.changes

- 签名
- 合规检查
- 上传到仓库
- 等等

Other

install文件

格式:

文件来源 安装目标

cat debian/install

helloword usr/bin/

建议复杂的可以在 rules脚本里面处理

override_dh_install

Other

包维护脚本

preinst postinst prerm postrm

- preinst用于安装文件前.
- postinst 用于处理安装后配置
- prerm 卸载前
- postrm 卸载后.

新手不建议使用.

dpkg-buildpackage

debian打包工具 (dpkg-dev)

安装

sudo apt install dpkg-dev

常用的

dpkg-buildpackage -rfakeroot -D -us -uc -sa -jauto

dpkg-buildpackage

参数:

-rfakeroot 在fakeroot环境里面运行

-D 进行编译依赖检查.

-us参数不对源码gpg签名.

-uc参数不对changes文件进行gpg签名

-sa 包含source源码包.

-jauto 自动选择并行编译的job数量

知识大纲

打包进阶

- 打包环境配置
- debuild
- pbuilder
- sbuilder
- cowbuilder
- dpkg-deb
- dput/dget
- 小技巧

打包环境配置

目标:干净的打包环境

- debootstrap
- 挂载设备
- chroot
- apt-get build-dep / mk-build-deps
- dpkg-buildpackage

打包环境配置

debootstrap

安装:

sudo apt install debootstrap

命令行:

[OPTION...] SUITE TARGET [MIRROR [SCRIPT]]

打包环境配置

debootstrap

常用参数

--no-check-gpg

--arch

--include

打包环境配置

debootstrap

常用参数

--no-check-gpg

--arch

--include

debuild

- debian常用的打包工具
- dpkg-buildpackage封装
- 供简单的干净环境进行打包
- 对包进行包检查
- 保存打包过程的日志

debuild 相当于 dpkg-buildpackage + lintian 组合

pbuilder

Personal Debian Package Builder

- 提供一个干净的编译构建环境
- 并对基础环境进行缓
- 保证每次打包的环境都是最基础的打包环境
- 自动解决打包依赖

pbuilder

create

创建一个干净的base环境.

实际上使用 debootstrap 创建一个环境,然后将环境压缩成base.tgz 备用

pbuilder create --mirror http://mirrors.ustc.edu.cn/debian/ --architecture amd64 --distribution jessie

pbuilder

update

更新基础环境apt-cache 数据

#更新base.tgz

pbuilder update

pbuilder

build

通过dsc文件编译软件包

#编译 helloword包

pbuilder build helloworld_0.1.dsc

pbuilder

login

chroot 到环境中（将自动挂载设备）

#修改base环境

pbuilder login --save-after-login

pbuilder

- buildresult 参数修改编译结果输出路径
- 可以通过pbuilderrc 文件进行客置化
- 通过hook扩展pbuilder的各个过程

sbuild

- sbuild 工具与pbuilder工具原理和功能差不多。
- 最大差异是pbuilder通过每次解压base.tgz 来保证环境的干净和稳定。sbuild不强制保证基础打包环境。
- sbuild初始化打包环境比较快。
- 一般sbuild多用于自动打包平台和工具调用。

cowbuilder

- COW + pbuilder
- 默认开启了pbuilder 的apt下载的cache
- 保留了pbuilder的易用性
- 可以替代pbuilder使用

dpkg-deb

deb操作工具

常用的参数:

-b, --build 将目录编译成deb包

-c, --contents 显示deb包中的内容

-x, --extract 解压deb中包的数据

-R, --raw-extract 全部解压包的内容

dpkg-deb

解压包数据

dpkg-deb -x helloworld_0.1_amd64.deb tmp

```
root@deepin#tree tmp/
tmp/
`-- usr
    |-- bin
    |   `-- helloworld
    `-- share
        `-- doc
            `-- helloworld
                |-- changelog.gz
                `-- copyright

5 directories, 3 files
```

dpkg-deb

解压所有内容

dpkg-deb -R helloworld_0.1_amd64.deb tmp

```
root@deepin#tree tmp/
tmp/
|-- DEBIAN
|   |-- control
|   `-- md5sums
`-- usr
    |-- bin
    |   `-- helloworld
    `-- share
        `-- doc
            `-- helloworld
                |-- changelog.gz
                `-- copyright

6 directories, 5 files
```

dget/dput

dget

通过dsc文件下载软件包的数据。

dput

将打包完成的deb推送到远端仓库中(临时数据
目录

小技巧

快速修改二进制包

- dpkg-deb -R xx.deb temp
- cd temp #修改包内容
- dpkg-deb -b . ../xxx.deb

小技巧

快速测试打好的包

- debi #安装包 dpkg -i
- debc # 显示包内容 dpkg -c

小技巧

打包版本对比

- debdiff
  - debdiff a1.deb a2.deb
  - debdiff a1.dsc a2.dsc
  - debdiff a1.changes a2.changes

小技巧

其他

- reprepro
- quilt
- git-buildpackage
- debcheckout
- alien

提示

参考资料：

- https://manpages.debian.org/jessie/devscripts/debuild.1.en.html
- https://pbuilder.alioth.debian.org/
- https://wiki.debian.org/sbuild
- https://wiki.debian.org/cowbuilder
- https://wiki.debian.org/DebianMaintainer

Thanks

Questions?

工作岗位

hr@deepin.com

深度科技官网