Hochschule Esslingen
University of Applied Sciences

HElikopter Project

Automotive Systems
Faculty Graduate School

Repository

Table of content

Problem

The program for the Helicopter shall be provided over a central Repository. The Repository includes the latest software version. The Helicopter shall be able to do software updates OTA. These updates can be implemented in the Helicopter in a mannor, where the update is made at boot. This implementation is out of scope in this document.

Implementation

1 Build Package

1: Decide on the name of your package. Standard debian notation is all lowercase in the following format:
<project>_<major version>.<minor version>-<package revision>

For example, you could name your first package...

```
helloworld_1.0-1
```

2: Create a directory to make your package in. The name should be the same as the package name.

```
mkdir helloworld_1.0-1
```

3: Pretend that the packaging directory is actually the root of the file system. Put the files of your program where they would be installed to on a system.

```
mkdir helloworld_1.0-1/usr

mkdir helloworld_1.0-1/usr/local

mkdir helloworld_1.0-1/usr/local/bin

cp "~/Projects/HelloWorld/helloworld" helloworld_1.0-1/usr/local/bin
```

4: Now create a special metadata file with which the package manager will install your program...

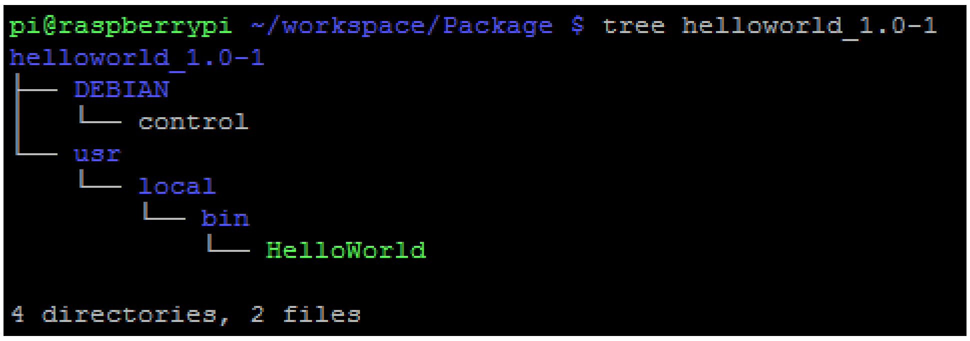
```
mkdir helloworld_1.0-1/DEBIAN

gedit helloworld_1.0-1/DEBIAN/control
```

Put something like this in that file...

```
Package: helloworld

Version: 1.0-1

Section: base

Priority: optional

Architecture: i386

Depends: libsomethingorrather (>= 1.2.13), anotherDependency (>= 1.2.6)

Maintainer: Your Name <you@email.com>

Description: Hello World

 When you need some sunshine, just run this

 small program!
```

(the space before each line in the description is important)

The tree architecture should look like that:



```
pi@raspberrypi ~/workspace/Package $ tree helloworld_1.0-1
helloworld_1.0-1
├── DEBIAN
│   └── control
└── usr
    └── local
        └── bin
            └── HelloWorld

4 directories, 2 files
```

5: Now you just need to make the package:

```
dpkg-deb --build helloworld_1.0-1
```

Output is a finished Debain Package with the file ending .deb

2 Upload Package

You can create your own repository with the tool **reprepro** on Linux

With **reprepro**, you can easily create and manage your own package repositories. The tool is located in the Ubuntu repositories and can be used to provide own packages or to mirror existing repositories. For situations where complete repositories are to be mirrored for provision in their own local network, there are alternatives such as apt-mirror.

Installation:
Reprepro can be installed under Linux over the repository

```
:~$ apt-cache search reprepro
reprepro - Debian package repository producer
:~$ sudo apt-get install reprepro
```

You can find useful information on how to set up a repos:

```
:~$ ls /usr/share/doc/reprepro/
changelog.Debian.gz  copyright  examples  FAQ.gz  manual.html  NEWS.gz
README  README.Debian  recovery  short-howto.gz  TODO
```

Configuration:
In the following example, a separate user with username “repo” was created to provide the packages in the home directory.

```
:~$ pwd
/home/repo
```

For the packages, a separate directory is created as well as a directory for the configuration of the repos:

```
:~$ mkdir packages
:~$ mkdir packages/conf
```

The first configuration step is to create the distribution file, which determines which distribution, architecture, etc., the repository is used for. Further examples of the distributions file can be found in (1) (wiki.debian.org) and (2) (debian-administration.org):

```
:~/packages$ vi conf/distributions
Origin: HelloWorld
Label: helloworld
Codename: precise
Suite: stable
Architectures: i386 amd64 source
Components: main
```

"Origin" and "Label" are descriptive fields which are copied into the release file. "Codename" is a mandatory field and specifies the unique identifier of the distribution (e.g., precise, quantal for Ubuntu, wheezy for Debian) (3) :
   Codename: This required field is the unique identifier.

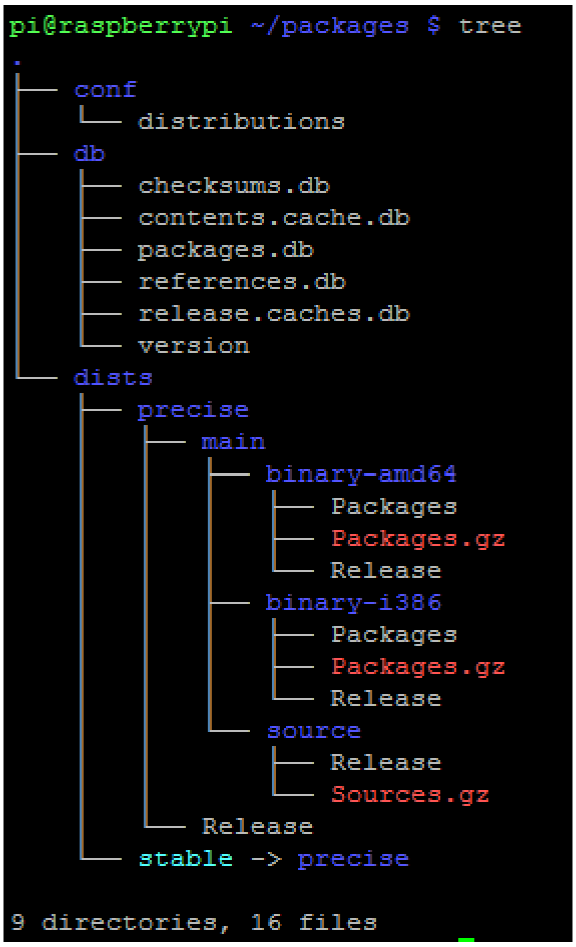
The "Suite" parameter specifies the common parameters such as "stable", "unstable" or "testing":
   Suite: This optional field is simply copied into the release files. In Debian it contains the names like stable, testing or unstable. To create symlinks from the Suite to the Codename, use the createsymlinks command of repre-pro.

"Architectures" and "Components" specify the target architecture of the system (32-bit, 64-bit, "source" for source code) as well as the components of the distribution (main, contrib etc.):
   Architectures: This required field lists the binary architectures

Include packages:
When using the "Suite" option in the "distributions" file, symbolic links are generated from "stable" to "precise" in the first step:

```
reprepro -b /home/repo/packages createsymlinks
```

This produces following folder structure (DB files are not incuded at initial set up, these are added when a repository is added):

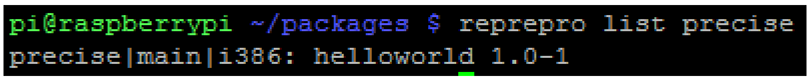


```
pi@raspberrypi ~/packages $ tree
.
├── conf
│   └── distributions
├── db
│   ├── checksums.db
│   ├── contents.cache.db
│   ├── packages.db
│   ├── references.db
│   ├── release.caches.db
│   └── version
└── dists
    ├── precise
    │   ├── main
    │   │   ├── binary-amd64
    │   │   │   ├── Packages
    │   │   │   ├── Packages.gz
    │   │   │   └── Release
    │   │   ├── binary-i386
    │   │   │   ├── Packages
    │   │   │   ├── Packages.gz
    │   │   │   └── Release
    │   │   └── source
    │   │       ├── Release
    │   │       └── Sources.gz
    │   └── Release
    └── stable -> precise

9 directories, 16 files
```

A first package is now added to the repository. This is made with the .deb file we created in step 1:

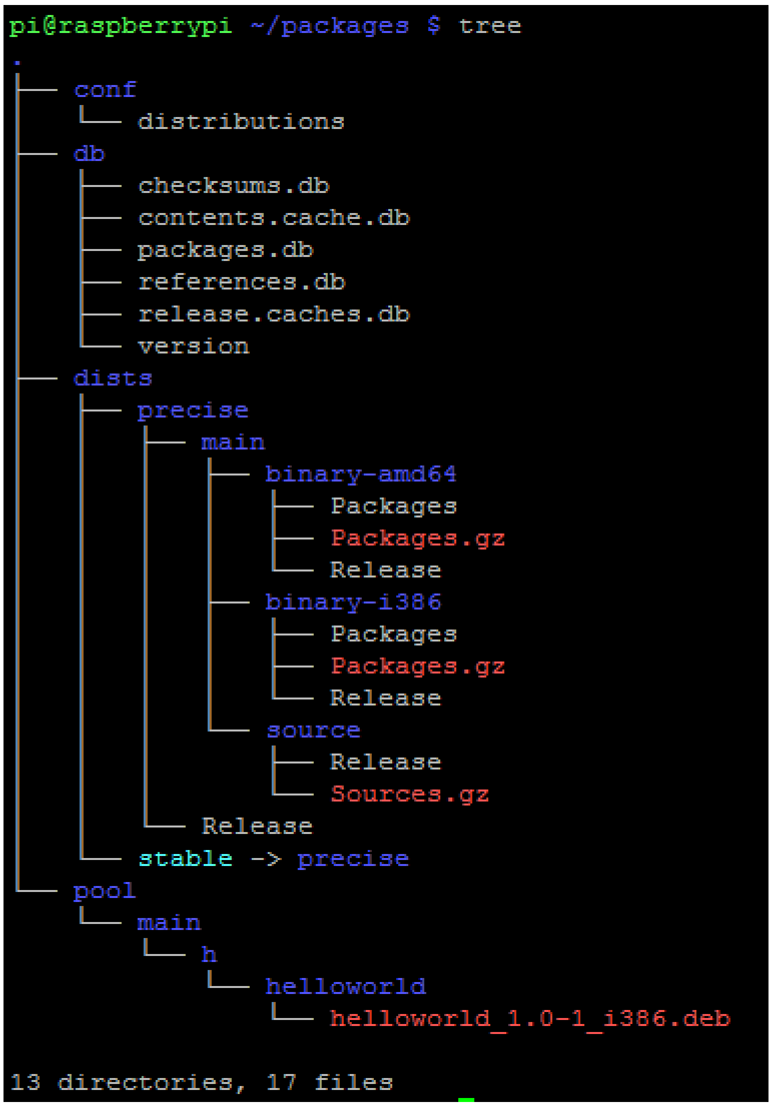
```
:~$ reprepro -V includedeb precise helloworld_1.0-1.deb
```

The contents of the repos are listed for checking purposes:



```
pi@raspberrypi ~/packages $ reprepro list precise
precise|main|i386: helloworld 1.0-1
```

The tree overview shows where the packages are located in the directory structure:



```
pi@raspberrypi ~/packages $ tree
.
├── conf
│   └── distributions
├── db
│   ├── checksums.db
│   ├── contents.cache.db
│   ├── packages.db
│   ├── references.db
│   ├── release.caches.db
│   └── version
├── dists
│   ├── precise
│   │   ├── main
│   │   │   ├── binary-amd64
│   │   │   │   ├── Packages
│   │   │   │   ├── Packages.gz
│   │   │   │   └── Release
│   │   │   ├── binary-i386
│   │   │   │   ├── Packages
│   │   │   │   ├── Packages.gz
│   │   │   │   └── Release
│   │   │   └── source
│   │   │       ├── Release
│   │   │       └── Sources.gz
│   │   └── Release
│   └── stable -> precise
└── pool
    └── main
        └── h
            └── helloworld
                └── helloworld_1.0-1_i386.deb

13 directories, 17 files
```

Remove Packages:
To remove packages, the remove command is called. The following example also restricts the removal to a specific architecture:

```
:~$ reprepro -A i386 remove precise helloworld
Exporting indices..
```

3 Distribute the package via Apache
There are several ways to distribute the repos via a web server. For all variants, the internal configuration files "/ conf" and "/ db" should be protected against unauthorized access. The following setup uses "mod userdir" to publish to the home directory of the user "repository". The first step is to activate the "userdir" module: (4)

```
:~$ sudo a2enmod userdir
```

For security reasons and because no other user should use "userdir", the userdir configuration file is modified as follows:

```
:~$ sudo vi /etc/apache2/mods-enabled/userdir.conf
<IfModule mod_userdir.c>
        UserDir public_html
        UserDir disabled
        UserDir enabled repository
[...]
```

We then created the public_html folder in the home directory of repository:

```
:~$ mkdir public_html
```

A symbolic link provides the packages in the public_html:

```
:~/public_html$ ln -s ../packages packages
```

A change in the file directories prevents access to the configuration folders:

```
:~/public_html/packages$ chmod 750 conf/
:~/public_html/packages$ chmod 750 db/
```

Create a “.list” file

The ".list" file simplifies adding the repos for the user: (5)

```
:~/public_html$ vi helloworld.list
deb http://192.168.0.125/~repository/packages precise main
```

The IP address must be adapted according to the correct hostname!

4 Include Package to local mirror list

A user can add and use the repo as follows:

```
:~$ wget http://192.168.0.125/~repo/helloworld.list
:~$ sudo mv hellworld.list /etc/apt/sources.list.d/
:~$ sudo apt-get update
:~$ apt-cache search helloworld
helloworld - a simple Hello World program
```

Artikel I.Literaturverzeichnis

1. apt-mirror. [Online] https://wiki.ubuntuusers.de/apt-mirror/.

2. reprepro. [Online] http://mirrorer.alioth.debian.org/.

3. CONFIG FILES. [Online] http://mirrorer.alioth.debian.org/reprepro.1.html#CONFIG%20FILES.

4. Apache. [Online] https://httpd.apache.org/docs/2.2/howto/public_html.html.

5. Debian repository. [Online] https://wiki.debian.org/DebianRepository/SetupWithReprepro?action=show&redirect=SettingUpSignedAptRepositoryWithReprepro.