Exercise sheet 3 – Software packages

Goals:

- Software management
- Create own `deb` packages
- Use and create `flatpak` packages

Exercise 3.1: Software management

(a) Search for the package `rar` in the package repository.

Proposal for solution: `apt search rar`

(b) Use the command `apt list rar`. What version of rar is inside the repository?

Proposal for solution: `apt list rar`
The version 2:5.5.0-1 of the package is inside the repository.

(c) Has the package `rar` any dependencies?

Proposal for solution: `apt depends rar`

```
Depends: libc6
Depends: libgcc1
Depends: libstdc++6
Suggests: unrar
```

(d) Install the `rar` package.

Proposal for solution: `sudo apt install rar`

(e) Use the commands `apt search rar` and `apt list rar` again. Are there any differences?

Proposal for solution: `apt search rar; apt list rar`
At the `search` command, the `p` was replaced with an `i`, and at the output of the second command, it is stated that the package is installed.

(f) Test if the `rar` program works.

Proposal for solution: `rar`
The help does appear

(g) Delete the `rar` package.

Proposal for solution: `sudo apt remove rar`

Exercise 3.2: Build your own deb package

This exercise is loosely based on
`https://wiki.ubuntuusers.de/Grundlagen_der_Paketerstellung`
Important: The steps have to be performed in exactly this order, otherwise you will run into troubles.

(a) Create a directory `hello`

Proposal for solution: `mkdir hello`

(b) Download the `hello-2.10.tar.gz` sources into the directory `hello` with:
`wget http://ftp.gnu.org/gnu/hello/hello-2.10.tar.gz`

Proposal for solution:

```
cd hello
wget http://ftp.gnu.org/gnu/hello/hello-2.10.tar.gz
```

(c) Extract `hello-2.10.tar.gz` with `tar -xf hello-2.10.tar.gz` and change into the newly created folder.

Proposal for solution:

```
tar -xf hello-2.10.tar.gz
cd hello-2.10
```

(d) Initialize the files for the debian package and change into the newly created folder.

Proposal for solution:

```
dh_make -f ../hello-2.10.tar.gz
cd debian
```

(e) Remove all files that are not needed (All files ending with `*.ex`, `*.EX` and the `README.*` files).

Proposal for solution: `rm *.ex *.EX README.*`

(f) Edit `changelog`:

- Set the stability to `bionic`. You can set your own comment behind the `*`.
- At the last line enter your name and e-mail address.

Proposal for solution:

```
hello (2.10-1) bionic; urgency=medium

  * First try of a deb package build.

 -- bs-dev <dev@unknown>  Tue, 21 Aug 2018 12:53:31 +0200
```

(g) Edit `control`:

- Set `Section` to `misc`.
- Set the `priority` to `optional`.
- At `Maintainer` enter your name and e-mail address.
- At the end of the file enter a description of the package.

Proposal for solution:

```
Source: hello
Section: misc
Priority: optional
Maintainer: bs-dev <dev@unknown>
Build-Depends: debhelper (>= 10), autotools-dev
Standards-Version: 4.1.2
#Homepage: <insert the upstream URL, if relevant>
#Vcs-Git: https://anonscm.debian.org/git/collab-maint/hello.git
#Vcs-Browser: https://anonscm.debian.org/cgit/collab-maint/hello.git

Package: hello
Architecture: any
Depends: ${shlibs:Depends}, ${misc:Depends}
Description: The classic greeting and a package build example
 The GNU hello program produces a familiar, friendly greeting.  It
 allows non-programmers to use a classic computer science tool which
 would otherwise be unavailable to them.
 .
 Seriously, though: this is an example of how to do a Debian package.
 It is the Debian version of the GNU Project's `hello world' program
 (which is itself an example for the GNU Project).
```

(h) Edit `copyright`:

- Set `Source` to `http://ftp.gnu.org/gnu/hello/hello-2.10.tar.gz`.
- At the copyright for the `Files: *` set the `Copyright` to `1992-2018 Free Software Foundation, Inc.` and the `License` to `GPL-3+`.
- At `Files: debian/*` set the `Copyright` to your name and your e-mail, at `License` set the license to `GPL-3+` and change in the following text all occurrences of 2 to 3.

Proposal for solution:

```
Format: http://www.debian.org/doc/packaging-manuals/copyright-format/1.0/
Upstream-Name: hello
Source: http://ftp.gnu.org/gnu/hello/hello-2.10.tar.gz

Files: *
Copyright: 1992-2018 Free Software Foundation, Inc.
License: GPL-3+

Files: debian/*
Copyright: 2018 bs-dev <dev@unknown>
License: GPL-3+
 This package is free software; you can redistribute it and/or modify
 it under the terms of the GNU General Public License as published by
 the Free Software Foundation; either version 3 of the License, or
 (at your option) any later version.
 .
 This package is distributed in the hope that it will be useful,
 but WITHOUT ANY WARRANTY; without even the implied warranty of
 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 GNU General Public License for more details.
 .
 You should have received a copy of the GNU General Public License
 along with this program. If not, see <https://www.gnu.org/licenses/>
 .
```

```
On Debian systems, the complete text of the GNU General
Public License version 3 can be found in "/usr/share/common-licenses/GPL-3".
```

(i) Go one directory up, configure the build (`./configure`), and build the package.

Proposal for solution:

```
cd ..                         #change into hello-2.10 folder
./configure                   #configure the build
dpkg-buildpackage -b -us -uc #build hello and the deb package
cd ..                         #change into hello folder
```

Exercise 3.3: Use and inspect your own package with dpkg

Alternatively: You can use the provided deb package from `OS_exercises/sheet_03_sw_package/hello_2.10-1_amd64.deb`.

(a) List information about the newly generated package.

Proposal for solution: `dpkg --info hello_2.10-1_amd64.deb`

(b) List all files which are inside the newly generated package.

Proposal for solution: `dpkg --contents hello_2.10-1_amd64.deb`

(c) Install the newly generated package into your system.

Proposal for solution: `sudo dpkg -i hello_2.10-1_amd64.deb`

(d) Check if the package is now installed (Hint: `dpkg -l` also accepts a pattern).

Proposal for solution: `dpkg -l hello`

(e) Run the program.

Proposal for solution: `hello`

(f) Search for all files named hello in all installed packages.

Proposal for solution: `dpkg -S hello`

(g) Remove the `hello` package.

Proposal for solution: `sudo dpkg -r hello`

Exercise 3.4: Flatpak software management

(a) List all installed flatpak packages.

Proposal for solution: `flatpak list`

(b) Search for packages containing “pdf”.

Proposal for solution: `flatpak search pdf`

(c) Install the `eu.scarpetta.PDFMixTool` package.

Proposal for solution: `flatpak install flathub eu.scarpetta.PDFMixTool`

(d) Run the installed program.

Proposal for solution: `flatpak run eu.scarpetta.PDFMixTool`

(e) Print information about the package.

Proposal for solution: `flatpak info eu.scrapetta.PDFMixTool`

(f) Remove the `eu.scarpetta.PDFMixTool` package.

Proposal for solution: `flatpak uninstall eu.scrapetta.PDFMixTool`

Exercise 3.5: Build your own flatpak package (optional)

(a) Follow the tutorial of `http://docs.flatpak.org/en/latest/first-build.html` and build your own flatpak package.

Proposal for solution: At first install the needed SDK (the runtime should be already installed):
`flatpak install flathub org.freedesktop.Sdk//1.6`
Then the application (here a script) has to be built:

```
#!/bin/sh
echo "Hello world, from a sandbox"
```

After that the manifest has to be written:

```
{
    "app-id": "org.flatpak.Hello",
    "runtime": "org.freedesktop.Platform",
    "runtime-version": "1.6",
    "sdk": "org.freedesktop.Sdk",
    "command": "hello.sh",
    "modules": [
        {
            "name": "hello",
            "buildsystem": "simple",
            "build-commands": [
                "install -D hello.sh /app/bin/hello.sh"
            ],
            "sources": [
                {
                    "type": "file",
                    "path": "hello.sh"
                }
            ]
        }
    ]
}
```

Now the package can be built and tested:

```
flatpak-builder build-dir org.flatpak.Hello.json
flatpak-builder --run build-dir org.flatpak.Hello.json hello.sh
```

Now put it into a local repository

```
flatpak-builder --repo=repo --force-clean build-dir org.flatpak.Hello.json
```

Now add the local repository to the available repositories and install the app

```
flatpak --user remote-add --no-gpg-verify tutorial-repo repo
flatpak --user install tutorial-repo org.flatpak.Hello
```

Now you can run the app with `flatpak run org.flatpak.Hello`

(b) Make your own notes on how to build a flatpak package.