FUSS - Developer's Manual

Release 12.0

FUSS Lab

Jul 14, 2023

Index

This manual is a guide to the maintenance and development of the FUSS GNU/Linux distribution based, server- and client-side on Debian 12 "Bookworm."

The most recent version of this manual can be read at https://fuss-dev-guide.readthedocs.io/en/latest/

CHAPTER 1

FUSS project guides

The fuss project guides (user, referent, tech, and dev) are written in reStructuredText and published using the sphinx documentation generator

1.1 Repository

Guide sources are in git repositories hosted on gitlab; having an account on the platform and a configured ssh key can be cloned with:

```
git clone git@gitlab.com:fusslab/fuss-user-guide.git
git clone git@gitlab.com:fusslab/fuss-referent-guide.git
git clone git@gitlab.com:fusslab/fuss-tech-guide.git
git clone git@gitlab.com:fusslab/fuss-dev-guide.git
```

otherwise you can use https access:

```
git clone https://gitlab.com/fusslab/fuss-user-guide.git
git clone https://gitlab.com/fusslab/fuss-referent-guide.git
git clone https://gitlab.com/fusslab/fuss-tech-guide.git
git clone https://gitlab.com/fusslab/fuss-dev-guide.git
```

1.2 Creation of new pages

To add a page to a guide:

- Create a new file with the extension `.rst`, e.g. `title-of-article.rst`; the file name should contain only lowercase letters, numbers and the hyphen `-`, with no accented letters, spaces or other special characters.
- Add the article to the *doctree* in `index.rst`, adding the file name without an extension (e.g. `title-of-article`) in the appropriate position in relation to the other articles.

1.3 Local Build

While you are making changes, it can be useful to generate html pages locally to get a preview of what will be published next.

1.3.1 Setup

The following dependencies must be installed for generation (on FUSS client, or a recent version of Debian or derivatives):

```
apt install python3-sphinx python3-sphinx-rtd-theme python3-stemmer
```

1.3.2 Build

To generate the html pages:

```
cd
<path/del/repository>/docs
make html
```

And you can then see the result with:

```
sensible-browser _build/html/index.html
```

1.4 Update published version

The version published on readthedocs is automatically updated whenever the branch is updated `master` on gitlab; if you have write access to the repositories, that is sufficient:

```
cd <path/del/repository>
git push origin master
```

(or simply `git push` if you are already on the `master` branch)

If you do not have write access to the repository, you can instead create a merge request

1.5 Style guidelines

1.5.1 Screenshots and terminal examples

For guides that refer to graphic programs, screenshots should be added.

Regarding examples from the terminal, better to report commands and output as text, for more accessibility, using code blocks, for example:

```
introductory text::

    $ command
    output of command
    # /sbin/command
    output of the command run by root
```

Note: Double colon in reStructuredText code is converted to a single colon in published versions, unless it is preceded by spaces (or in a paragraph by itself), in which case it is removed.

1.5.2 Structure levels

reStructuredText allows different fonts to be used for different levels of structure in a document, as long as they are consistent within the individual file.

However, it is best to follow the convention of Python's Style Guide for documenting, which states:

- # with double line for parts (not used in these manuals);
- * with double line for chapters (corresponding for us to files);
- = for sections;
- - For subsections;
- ^ for subsections;
- " for paragraphs.

1.6 Tips and tricks

1.6.1 Migration from Redmine wiki

Documents on the redmine wiki use the (source) textile format; to convert it to reStructuredText, the program Pandoc can be useful

Once you have copied the page source to a local `article.txt` file you can use the following command to get a version of it in reStructuredText in the `article.rst` file:

```
pandoc -f textile -t rst -o article.rst article.txt
```

Such a file is a good starting point, but it is not ready for direct inclusion in FUSS guides without first manually verifying the contents; in particular, it will definitely be necessary to manually fix some features.

- Structure levels will need to be adjusted to match the standard specified above, paying attention to where the document is placed (e.g., if you are converting a wiki page to a section or subsection of a help page).
- Images are uploaded from the original file on the wiki; however, for future maintainability it is appropriate to upload them locally within the guide.

  After you have downloaded the images, say in the `images/article/` directory, and moved the pandoc-generated references from the end of the document to the location where the image is to appear, you can convert them to `figure` directives with the following vim command:

```
:%s/|image.*| image:: https:\/\work.fuss.bz.it/attachments/download\/.*//
↪figures:: images/article//.
```

- pandoc generates blocks of code introduced by a line containing only `::`, even when the previous paragraph ends with `::`; for better elegance and readability of the source these can be converted by putting `::` only on the previous paragraph.

  However, both cases are valid reST constructions that are compiled in a similar presentation.

- Some wiki articles contain sections with indications such as "caution" or "note": in these cases it may be worth converting them to the relevant reStructuredText specific directive

1.6.2 Screenshot

Resize a window accurately

To take screenshots of a program window, it is useful to resize it to the precise size you want to give the screenshot; you can use the following command to do this:

```
sleep 3 ; xdotool getactivewindow windowsize 1024 768
```

which waits 3 seconds, during which time you can change the active window from the terminal to the desired window, and then performs the resize.

CHAPTER 2

Packages and Repositories

The FUSS distribution includes a repository of additional packages to the base (Debian), available at https://archive.fuss.bz.it/ and hosted at `isolda.fuss.bz.it` in the `/iso/repo` directory.

Index

2.1 Build packages

In the repositories of the software developed for FUSS is the `debian` directory containing the files necessary for generating `.deb` packages.

In sufficiently recent projects, such a directory is present only in a dedicated branch, with a name like `fuss/<version>`; for these cases see also the section *Verifying versions in the repository* .

2.1.1 Setup

To perform local builds of packages, some development tools must be installed:

```
# apt install devscripts dput-ng dh-systemd dh-python
```

Note: Make sure you have also installed package Recommends (this is the default behavior of `apt`, unless you have disabled it manually), particularly in the case of `dput-ng`.

In addition, it is necessary to set the `DEBEMAIL` and `DEBFULLNAME` environment variables, containing the developer's full name and email, respectively, which will be used to update some metadata.

To use `dput-ng` to make uploads you need to configure it by creating the file `~/.dput.d/profiles/fuss-<version>.json` containing:

```
{
    }, "method": "sftp",
    "fqdn": "archive.fuss.bz.it",
    "incoming":"/iso/incoming/<version>",
    "allow_dcut": false,
    "allowed-distribution": {},
    "codenames": null,
    "post_upload_command": "ssh -S none isolda.fuss.bz.it 'sudo /iso/bin/post-.
→upload'",
    "hooks": [
        "allowed-distribution",
        "checksum",
        "suite-mismatch"
    ]
}
```

where `<version>` is `bookworm` and `bookworm-proposed-updates` for the current version of FUSS ser- ver and client, and `buster` and `buster-proposed-updates` for the legacy server version and `bullseye` and `bullseye-proposed-updates` for the client version.

Tip: In some versions of dput-ng there is a bug, #952576 whereby any `post-upload` errors are not reported, making the failure silent. If you run into that situation, a possible workaround is to replace the `post_upload_command` configuration with the following:

```
"post_upload_command": "ssh -S none isolda.fuss.bz.it '/iso/bin/post-upload 2>&1'",
```

so that any errors are redirected to stdout and are displayed.

Also make sure that you can access `archive.fuss.bz.it` via ssh without additional options; for example, you may need to add the following to `~/.ssh/config`:

```
Host archive.fuss.bz.it
    User root
```

Note: dput-ng uses paramiko to make ssh connections; this implies that options set directly in `~/.ssh/config` are read correctly, but there is no support for using `Include` to split the configuration over multiple files.

In addition, the fingerprint of servers will not be saved, but will be asked each time to

verify it. As of March 2019, the fingerprints of isolda are:

```
256 SHA256:aLTgA+Trj5iYo0dl0i8Q82aigs3K/dPwDbazrvG95YY root@isolda (ECDSA)
256 SHA256:7i6j0jXPWRW6LXDGbR+HWr3AFJi6gGSmdW41uBRJV4 root@isolda (ED25519)
2048 SHA256:OkP1maDf0pSIGCdq1mph8oI8CTADMrFXfe3aty608SA root@isolda (RSA)

256 MD5:b1:a1:ec:cb:a5:39:c8:8d:39:f1:dd:ba:aa:be:38:11 root@isolda (ECDSA)
256 MD5:21:41:8b:19:1b:25:b5:9c:f2:5c:e8:b9:8b:08:07:f8 root@isolda (ED25519)
2048 MD5:bd:88:bd:5f:bc:52:03:0b:88:d9:0c:2b:86:59:dc:92 root@isolda (RSA)
```

Cowbuilder

Note: cowbuilder and pbuilder are tools for managing chroots within which to perform package builds in a clean environment that is fairly isolated from the base system.

Building packages within an isolated system is useful to avoid influences from one's own system (with libraries and other dependencies already installed, perhaps in non-standard versions), but it is also convenient in case one vo- les to generate packages for distributions other than those in use (e.g., build for buster on a bookworm system)

In addition to the above, install `cowbuilder`, `pbuilder` and `git-buildpackage`:

```
# apt install pbuilder cowbuilder git-buildpackage
```

and make sure that the user you want to use to launch builds is part of the `sudo` group.

Then create the base chroots for the distributions currently (January 2023) in use buster, bullseye and bookworm:

```
# cowbuilder --create --distribution buster --debootstrap debootstrap \.
  --basepath /var/cache/pbuilder/base-fuss-buster.cow
# cowbuilder --create --distribution bullseye --debootstrap debootstrap \.
  --basepath /var/cache/pbuilder/base-fuss-bullseye.cow
```

(continues on next page)

(continued from previous page)

```
# cowbuilder --create --distribution bookworm --debootstrap debootstrap \.
  --basepath /var/cache/pbuilder/base-fuss-bookworm.cow
```

Tip: cowbuilder can also be used under debian-derived distributions, such as ubuntu; however, in this case you must explicitly specify the use of a debian mirror by adding options to the commands above:

```
--mirror http://<a valid debian mirror>/debian --components main
```

additionally, for operation you need Debian's GPG signing keys, which are typically installed with:

```
apt-get install -y debian-archive-keyring
```

Add the backports and fuss repositories to the newly created base chroots: copy the key file to the chroot:

```
# wget -O \
  /var/cache/pbuilder/base-fuss-bookworm.cow/usr/share/keyrings/fuss-keyring.gpg \
  https://archive.fuss.bz.it/apt.gpg
```

Then log into the chroot:

```
# cowbuilder --login --save-after-login \
  --basepath /var/cache/pbuilder/base-fuss-bookworm.cow
```

and make the changes to `/etc/apt/sources.list` (replacing `<mirror>` with an appropriate debian mirror, such as the one already in `/etc/apt/sources.list`:

```
# echo 'deb <mirror> bookworm-backports main' >> /etc/apt/sources.list
# echo 'deb [signed-by=/usr/share/keyrings/fuss-keyring.gpg] http://archive.fuss.
↪bz.it/ bookworm main contrib' \
  >> /etc/apt/sources.list
# apt update
# exit
```

Until the buster version, the instructions were slightly different:

```
# echo 'deb <mirror> buster-backports main' >> /etc/apt/sources.list
# echo 'deb http://archive.fuss.bz.it/ buster main contrib' \
  >> /etc/apt/sources.list
# apt install gnupg
# apt-key add - # paste the contents of
                # https://archive.fuss.bz.it/apt.key followed by ctrl-
d # apt remove gnupg
# apt autoremove
# apt update
# exit
```

Note: in the minimal chroot from stretch onwards, there is no gnupg, which is required for the use of apt-key add: we install it for the task and remove it immediately afterwards to make sure that the image is always minimal and continue to notice any unspoken dependencies in the packages we generate.

Repeat the same for any other chroots.

In case the chroots have been created for some time it is advisable to update them, with the following commands:

```
# cowbuilder --update --basepath /var/cache/pbuilder/base-fuss-buster.cow/
# cowbuilder --update --basepath /var/cache/pbuilder/base-fuss-bullseye.cow/
# cowbuilder --update --basepath /var/cache/pbuilder/base-fuss-bookworm.cow/
```

2.1.2 Clone and/or update the repository

Clone the desired project repository:

```
$ git clone https://work.fuss.bz.it/git/<project>
```

In old projects the branch to be built for upload is `master`, in recent ones `fuss/*` (`fuss/master` or other depending on the target), in both cases to be updated in case you already have a local clone of the repository:

```
$ git checkout [fuss/]master
$ git pull
```

In case you are doing a release for a recent package (packaging in `fuss/*`) remember to update the branch with the changes you want to integrate into the release.

Hint: For example, to make a release of the active development version of a recent package, you start by making sure you have the current version of the development branch, `master`:

```
$ git checkout master
$ git pull
```

you possibly check that the version number in `setup.py` is correct, and eventually update and commit it:

```
$ $EDITOR setup.py
$ git add -p setup.py
[...]
$ git commit -m 'Bump version to <version>'
$ git push
```

you move on to the branch with the packaging, making sure that it is also up to date:

```
$ git checkout fuss/master
$ git pull
```

And you merge the development branch:

```
$ git merge master
$ git push
```

at this point we proceed as described in the next point by updating the package changelog, etc.

2.1.3 Versionamento

In order to publish the package, you must increment the version number in the `debian/changelog` file.

The version number to be given depends on the type of package, as described in the *Versioning Policy* section and in the development guides of the specific packages, but in most cases it will be to increase the patch level (e.g., from 10.0.5-1 to 10.0.6-1).

Note: In packages containing programs in python it is generally necessary to keep the version number updated in `setup.py` as well; as with debsrc above this should be mentioned in the README of the packages.

Note that updating the version number in `setup.py` should be done in the development branch, not the packaging branch.

The `dch` program, allows you to automate the editing of the `debian/changelog` file that contains the package version.

- When you start making changes use the command `dch -v <new_version>` to create a new room and open the changelog in the default editor.

  The required version number and the special release `UNRELEASED` will be set, indicating that the changes are still being worked on.

  You can also use `dch` without options: that way if the last stanza results `UNRELEASED` the file will be opened as is, while if the last stanza reports a release as `unstable` a new one is created by incrementing the version number.

  Beware that in the latter case dch may not be able to guess the correct version: check and if necessary correct. Also, in case you are not listed among Maintainers and Uploaders in `debian/control` a `Non Maintainer Upload` line will be added which is not relevant to us and should be removed.

  In case more than one person makes changes, dch will divide them into sections headed with the name of the person who made the change.

- Describe the changes made, possibly indicating the reference tickets from which the change requests originate.
- As you make changes, describe them if necessary in the changelog, using `dch` without options, as described above.
- When you are ready to publish the package, edit `UNRELEASED` with `unstable` in the first line; this can also be done with the `dch -r` command.

2.1.4 Checking repository status and push

Before doing the build, make sure that you have committed all the changes you have made, that you have no spurious files, and that you are on the correct branch (`master` or `fuss/*` depending on the age of the project), for example with the command:

```
$ git status
```

Then commit any remaining changes, taking care to keep development and packaging changes separate, indicating if possible in the commit log the ticket number associated with the change, with the words "refs #NUMBER."

```
$ git add -p <edited files>.
$ git add <files added>
$ git commit -m "<changes made>. refs #NumeroTicket"
```

Also either right before or right after the build, but before the upload, it is important to push those commits, so as to be sure that no conflicts with others' commits occur in the meantime:

```
$ git push
```

2.1.5 Build

Some packages, such as octofussd require prior creation of the original source tar, as spe- cified in the README of the respective repositories; in such a case, before running the previous command, it is necessary to run, in the root directory of the repository:

```
$ debian/rules debsrc
```

At this point you can use `pdebuild` to run the package build within an appropriate chroot managed by cowbuilder (replacing `bookworm` with `buster` or `bullseye` in appropriate cases):

```
$ DIST=bookworm pdebuild --use-pdebuild-internal --pbuilder cowbuilder -- --.
↪basepath /var/cache/pbuilder/base-fuss-bookworm.cow/
```

`pdebuild` will independently install the necessary dependencies within the chroot and perform the build.

Generally, the build infrastructure[1] is able to tell from the version number and other clues whether or not it is necessary to include the `.orig` tarball among what is to be uploaded. However, in the case where a backport is being performed this is not automatic: for the first backport of a certain upstream version it is necessary to provide for the inclusion of the source tarball with the `--debbuildopts "-sa"` option, i.e.:

```
$ DIST=bookworm pdebuild --buildresult ../build/ --use-pdebuild-internal --
↪pbuilder cowbuilder --debbuildopts "-sa" -- --basepath /var/cache/pbuilder/base-.
↪fuss-bookworm.cow/
```

Note: At the end of the build you will receive a warning `cannot create regular file /var/cache/ pbuilder/result/<package name>_<version>.dsc`; this is irrelevant and the necessary files that were generated are in the directory above the one from which `pdebuild was` launched.

Build with git-buildpackage

As an alternative to using `pdebuild` directly, but only for projects whose repository supports it through the use of a separate branch for packaging, it is possible to use `git-buildpackage` (or `gbp`): in addition to effecting the build in a minimal chroot this also ensures that there are no differences between what is committed (locally) and what is used for the build.

To perform the build in this case, it is necessary to move to the packaging branch, for example:

```
$ git checkout fuss/master
```

possibly report in that branch the changes made on master:

```
$ git merge master
```

And then you can build, in the general case with:

```
gbp buildpackage --git-pbuilder --git-debian-branch=fuss/master \
   --git-dist=fuss-bookworm --git-no-pristine-tar
```

To force the inclusion of the backports source tarball, as explained above, in this case simply add `-sa`.

2.1.6 Test

Through `apt install ./<filename>.deb` you can" install and test the package. Note the use of `./` to specify a local file.

Another useful command is `dpkg -c <filename>.deb` to check the files in the package.

[1] Specifically, the inclusion or non-inclusion of the source tarball is decided and carried out by `dpkg-genchanges`, invoked by `dpkg-buildpackage` to which `pdebuild` passes the options specified with the `--debbuildopts` parameter.
Generally this is done automatically, without needing to worry about who does what.

lintian

A very detailed diagnostic tool is `lintian`, which analyzes generated packets for various types of problems and launches with:

```
lintian --pedantic -Iiv <package>.changes
```

A limitation of this tool is that it is based on Debian standards, and in some cases errors may be false positives for fuss standards.

In particular, the following tags can be ignored.

- `changelog-should-mention-nmu`
- `source-nmu-has-incorrect-version-number`

and others that will later be added to this list.

2.1.7 Upload

To upload the package built with `dput-ng`, simply use the command:

```
$ dput fuss-<version> package_name_arch.changes
```

In case you want to proceed manually instead, you can copy the generated files on `isolda` to the directory `/iso/incoming/<version>` and update the repository with the command:

```
# /iso/bin/post-upload
```

Then check that no spurious files are left in `/iso/incoming/<version>`, and if so, delete them by hand.

2.1.8 Tagging

At the time when everything is ready for an upload, tag the commit corresponding to what will be uploaded with the command:

```
$ git tag -s -m 'Fuss release <version>' fuss/<version>.
```

this way the tag will be signed with its default gpg key; to not sign the tag:

```
$ git tag -a -m 'Fuss release <version>' fuss/<version>.
```

Remember to push the tags to the server:

```
$ git push --tags
```

2.2 Build packages in chroot

In case there are problems with using cowbuilder, it is also possible to use a simple chroot within which to install the build tools and clone the package.

2.2.1 Setup

To create a chroot and install the basic tools:

```
# mkdir <version>_build
# debootstrap <version> <version>_build (<mirror>) #
chroot <version>_build
# apt install debhelper devscripts dpkg-dev
```

where `<version>` is currently (January 2023) `bookworm` for current FUSS server and client, `buster` for the legacy version of FUSS server, and `bullseye` for that of FUSS client.

2.2.2 Build

Once you have cloned the repository (inside the chroot), incremented the version number as above and possibly generated the source tar, to run the package build run, in the root directory of the repository:

```
# dpkg-buildpackage -us -uc
```

If the procedure is successful, in the top directory you will find the generated `.deb` packages, and also the files `.changes`, `.dsc`, and `.tar.gz` with the package source.

The procedure may fail with an error containing:

```
Unmet build dependencies: <package1> <package2>
```

in which case simply install the packages and try again. These dependencies are listed in the `Build-Depends` field of the `debian/control` file, in case you want to make sure you already have them installed before building.

At this point you can proceed with testing, commit+push and upload as in the general case.

2.3 Versioning policy

2.3.1 Software developed for FUSS

For packages developed specifically for FUSS, there may be specific policies stated in the relevant developer guide and/or project READMEs.

In general, the pattern used is that the major version corresponds to the version of fuss for which the package is released (which in turn corresponds to the version of debian on which it is sufficient). Thus a package for FUSS 9 will have version like 9.X.Y, one for FUSS 10 10.X.Y and so on.

Packages may or may not be native: in the former case the version number is of the type 10.X.Y both for the package and in `setup.py`, while in the latter a revision number is added, e.g., 10.X.Y-Z; the latter should be incremented when the new version has changes in the packaging (i.e., in the debian directory), but not in the code.

Native packages must also have `3.0 (native)` in the `debian/source/format` file, while non-native pac- chets must have `3.0 (quilt)`, and to build them, a source tarball (`<name>_<10.X.Z>.orig.tar.gz`) must be generated, such as via `debsrc`.

2.3.2 Rebuild of debian packages

For packages taken from debian and rebuilt by us we follow a convention similar to that used by backports by adding `~fussN-X` to the version number, where N is the version of FUSS for which we are preparing the package and X is the backport revision.

If we do a rebuild of a package that for example has version 1.2.3-4 our version will be 1.2.3-4~fuss10+1 (+2 for a later rebuild with changes to packaging only, etc.).

2.4 Repository configuration

The `/iso/repo/conf/distributions` file defines the distributions used in the repository, with configuration snippets such as:

```
Origin: FUSS
Label: FUSS
Suite: bookworm
Codename: bookworm
Version: 10.0
Architectures: i386 amd64 source
Components: main contrib
Description: FUSS 10.0
SignWith: C00D47EF47AA6DE72DFE1033229CF7A871C7C823
```

In addition, previous and future versions of the distribution are defined in the same file. At present the configuration covers up to Debian version 12 (codename `bookworm`).

The various distributions can be reached from apt using, in `/etc/apt/sources.list`:

```
deb http://archive.fuss.bz.it CODENAME_DISTRIBUTION main contrib.
```

and the key with which the repository is signed can be installed on a debian or derivative machine by running, as root:

```
# wget -qO /usr/share/keyrings/fuss-keyring.gpg https://archive.fuss.bz.it/apt.key
```

2.4.1 Adding new distribution and/or new repository

In addition to the `/iso/repo/conf/distributions` file to indicate the new distribution and/or new repository, you need:

- Create a folder for package incoming spool in `/iso/incoming/<new distribution>.`
- Add the description and path related to the previous point in the file `/iso/repo/conf/incoming`
- Add to the `/iso/bin/post-upload` script the execution of processing the new inco- ming path. In this script, those no longer used should be removed when it is certain that there will no longer be new packages for a specific distribution.

2.4.2 Copying packages between different distributions

reprepro allows you to copy packages already loaded for one distribution to a different distribution; this is useful, for example, to take a package from `<distro>-proposed-updates` to `<distro>` once you have ascertained that it works properly.

The command is:

```
root@isolda:/iso/repo# reprepro copy bookworm bookworm-proposed-updates <name_
↪package>.
```

where care must be taken that the *first* distribution specified is the target distribution, followed by the source distribution.

2.4.3 Verification of versions in the repository

To know what versions of a package are in what distribution just use the command:

```
reprepro ls <package name>
```

2.5 Packaging managed with git

As described in the instructions, one of the conventions in use in Debian for git-based packaging has been adopted in more recent projects.

- The development branches of the project, including `master` do not contain the `debian` directory, as per the recommendation of the Debian UpstreamGuide.
- Branches whose names begin with `fuss/` contain the debian directory; generally the branch used for uploads of the current version will be `fuss/master`.
- On each release, the `master` branch is merged into `fuss/master` (but *never* the other way around) and the package can be generated using the methods described above.

In case you want to build with `gbp` (`git-buildpackage` the command to use will be:

```
gbp buildpackage \
--git-pbuilder \
--git-no-pristine-tar \
--git-debian-branch=fuss/<version> \
--git-dist=fuss-bookworm
```

adding `--git-export=WC` to make test builds of the current state of the working directory (instead of the-state at last commit) or `--git-ignore-new` to make a build corresponding to the last commit, ignoring any changes that may be present.

2.6 Standard configurations and new packages

2.6.1 Maintainer

The `debian/control Maintainer` field should have the value `FUSS team <packages@fuss.bz.it>`, so as to indicate that the package is maintained by a team.

2.7 Special packages

2.7.1 coova-chilli

The coova-chilli package on archive.fuss.bz.it is generated from our own repository https://gitlab.fuss.bz.en/fuss/coova-chilli/ copy of the upstream repository https://github.com/coova/coova-chilli.git to which we have added some packaging changes.

Specifically, for the 1.6 release there is a `1.6-patched` branch based on the 1.6 upstream tag with the addition of the `--enable-wpad` compile option, which is necessary for proper fuss-server operation.

To make new builds of version 1.6, it is therefore necessary to use the `1.6-patched` branch, merging any desired upstream changes into it; using `git-buildpackage` will have to be used:

```
$ gbp buildpackage --git-debian-branch=1.6-patched [--git-pbuilder]
```

For later versions, a similar branch should be created from the upstream tags, reporting any changes still needed.

Other branches in our repository contain packaging for earlier versions of FUSS, which are of historical utility only.

2.7.2 gspeech

The gspeech package present on archive.fuss.bz.it is derived directly from the packaging present on the upstream re- pository, from which to take any updated versions. Only a room with our version number has been added, like:

```
gspeech (0.9.2.0-1) buster; urgency=medium

  * Rebuild for FUSS 10

 -- Elena Grandi <elena@truelite.it> Tue, 02 Feb 2021 13:42:20 +0100
```

taking care to increase the version number from previous ones.

2.8 See also.

- https://wikitech.wikimedia.org/wiki/Reprepro#HOWTO

CHAPTER 3

Notes on Git

The development of fuss is managed through git repositories published on the gitlab instance https://gitlab.fuss.bz.it. For the basics of using git refer to the links in the *See also* section, but this document contains guidance on how to perform some specific tasks.

Index

3.1 Branch to support multiple distributions

Development in the `master` branch always refers to the most recent of the distributions supported for that package; if older distributions are also supported such support occurs in a branch named after the name of the distribution (e.g., `buster`).

Warning: In some packages the branch name is still in the form `fuss/<distribution>`: this is to be avoided, because it conflicts with the branch name used for packaging in a separate branch, but still has not been fully unified.

3.1.1 Changes to be applied to all distributions

In the case of making changes that need to be present in all versions, the good practice is to make the change in the most recent version (`master` branch), commit it, and then cherry pick the commit on the branch of the older versions.

For example, being in the following situation:

```
fuss-foo (master)$ git log --graph --abbrev-commit --pretty=oneline
* 7bc3fd2 (HEAD -> master) Start working on bullseye
* 0f46c37 (buster) Readme for the fuss-foo project
```

You make a change, commit it with the command:

```
fuss-foo (master)$ git commit -m 'Add project description.'
```

And you find yourself in the following situation:

```
fuss-foo (master)$ git log --graph --abbrev-commit --pretty=oneline
* 36c12d2 (HEAD -> master) Add project description.
* 7bc3fd2 Start working on bullseye
* 0f46c37 (buster) Readme for the fuss-foo project
```

At this point we switch to the branch of the old distribution, and `cherry-pick` the commit, that is, we copy it to the new branch:

```
fuss-foo (master)$ git checkout buster
fuss-foo (buster)$ git cherry-pick
36c12d2 Auto-merging README.rst
CONFLICT (content): Merge conflict in README.rst
error: could not apply 36c12d2... Add project description.
hint: After resolving the conflicts, mark them with
hint: "git add/rm <pathspec>", then run
hint: "git cherry-pick --continue".
hint: You can instead skip this commit with "git cherry-pick --skip".
hint: To abort and get back to the state before "git cherry-pick",
hint: run "git cherry-pick --abort".
```

Unfortunately, all did not go well and the change does not apply cleanly; we open the `README.rst` file and find the following:

```
Fuss FOO
========

<<<<<<< HEAD
This is FOO for buster
=======
This is FOO for bullseye

It is an example project.
>>>>>>> 36c12d2 (Add project description.)
```

We resolve the conflict manually, bringing the content to be:

```
Fuss FOO
========

This is FOO for buster

It is an example project.
```

and finally we continue the `cherry-pick` as suggested by the error message:

```
fuss-foo (buster)$ git add README.rst
fuss-foo (buster)$ git cherry-pick --continue
[...]
[buster c21a51e] Add project description.
 Date: Fri Jun 10 11:03:35 2022 +0200
 1 file changed, 2 insertions(+)
```

Confirming the commit message in the editor that opened. At

this point the situation has become:

```
fuss-foo (buster)$ git log --graph --abbrev-commit --pretty=oneline --all
* c21a51e (HEAD -> buster) Add project description.
| * 36c12d2 (master) Add project description.
| * 7bc3fd2 Start working on bullseye
|/
* 0f46c37 Readme for the fuss-foo project
```

Which is what we wished for.

Note: In most cases git is quite clever and it is rare to be in the situation above where cherry-picking requires manual intervention to resolve conflicts, except in one specific case: the `debian/ changelog` file.

If you keep the debian packaging inside a separate branch this is not a problem, but if you still have the packaging along with the code, you are in a situation where there is no good solution.

If you put the changes to `debian/changelog` in the same commit where the changes were made you get a cleaner repository history, but you are sure to find yourself having to resolve a conflict at cherry-pick time.

Committing `debian/changelog` separately instead prevents those conflicts, but leaves a dirtier history, with at- least two commits for each change: this is not a big deal when a changelog line corresponds to perhaps a dozen commits of a major development, but when a changelog line corresponds to one simple commit, as is often the case, it can be annoying.

3.2 See also.

- Pro Git by Scott Chacon book searchable online.
- Git Happens by Jessica Kerr video of an introductory talk on git.

CHAPTER 4

Metapackages

The FUSS distribution includes a number of metapackages to simplify the installation of educational or otherwise useful programs in the school environment, managed in the fuss-software project

4.1 Repository

The repository of metapackages can be cloned with:

```
$ git clone https://work.fuss.bz.it/git/fuss-software
```

`The master` branch refers to the latest release of FUSS; earlier versions are in the branches named with the codename of the related distribution.

4.2 Modification of metapackages

The files in the metapackages folder refer to the metapackage of the same name and contain the dependencies, one per line and preferably in alphabetical order.

Warning: Dependencies must be present within the configured repositories, otherwise the package will no longer be installable.

As of July 2018, this means that the packages you want to install must be present in Debian 9 "stretch" in the `main` and `contrib` sections.

4.2.1 Version number

The major version of metapackages must reflect the version of the Debian distribution being used; for example, a package for the "stretch" version will have 9 as its major version.

The patch level should be increased by one with each release, as usual.

The metapackages are native, so there does not have to be a debian version, only the three MAJOR.MINOR.PATCH components.

For package build and upload of changes see *Packages and Repositories*

CHAPTER 5

FUSS Server

`fuss-server` is a python script that launches an ansible playbook that configures a machine to run as a server in a FUSS network.

5.1 fuss-server

The `fuss-server` script is written to be compatible with python 2 and 3; the moment ansible switches to using python 3 you can eliminate python 2 compatibility.

The various subcommands correspond to functions of the same name and generally end with the `os. execvp` of a shell command to launch ansible; note that this terminates the execution of the python program, any subsequent code is not executed.

5.2 Playbook

Ansible is called with one of the following playbooks, depending on the subcommand used:

`create.yml` to configure a fuss-server from scratch.

`upgrade.yml` to update the configuration of a fuss-server.

`captive_portal.yml` to apply the additional configuration needed on captive portals.

`purge.yml` to purge the fuss-server configuration.

The latter restores some configuration files from backups, the others do not perform actions directly, but retrieve roles from the `roles` directory, so that code can be shared, particularly between `create` and `upgrade`.

5.3 Debian packages

The repository involves the generation of two .deb packages, `fuss-server` and `fuss-server-dependencies`; the former contains the actual fuss-server, while the latter is a metapackage that depends on all the packages installed by the ansible playbook.

fuss-server-dependencies is not required to use fuss-server, but is added for convenience to pre-install (and especially pre-download) all necessary packages.

For instructions on how to build packages and upload them to archive.fuss.bz.it you can see the article *Packages and Repositories*

5.3.1 Version numbers

The `fuss-server` package is native, so the version number is of the type X.Y.Z where X is the corresponding debian version number (e.g. 8 for jessie, 9 for stretch, 10 for buster).

5.3.2 Addictions

Some of the dependencies of the fuss-server package, particularly ansible (in the required version) and python-ruamel.yaml are available only in `jessie-backports`; to install the package you must have enabled that repository, and to use it you must also have the FUSS repository.

CHAPTER 6

FUSS Client

`fuss-client` is a python script that launches an ansible playbook that configures a machine as a client in a FUSS network.

6.1 fuss-client

The `fuss-client` script is written for python 3.

The `-a | -U | -r | -l` options are mutually exclusive and correspond to the `add`, `upgrade`, `remove`, and `listavail` methods, respectively; with the exception of the latter they end with the "`os.execvp` of a shell co-mandate to launch ansible; note that this terminates the execution of the python program, any subsequent code is not executed.

Before configuration, the `-a` option searches for and contacts a fuss-server (methods `_test_connection` and `_get_cluster`) to add the current machine to a cluster, via the octofuss api.

Note that there is no corresponding api for removing a machine from a cluster, an operation that must be done server-side.

The next step is the generation of a kerberos key for the client: this operation is performed on the server by the `add_client_principal` script, invoked via ssh, then the key is copied locally tra- mitted by scp. For authentication on the server, several cases are supported: access as root, access as a user with sudo permissions, or access with a key with permissions limited to only the operations required for the script.

6.2 Playbook

Ansible is called with one of the following playbooks, depending on the subcommand used:

`connect.yml` to configure a fuss-client

`remove.yml` to remove the fuss-client configuration.

The latter restores some configuration files from backups, the former does not perform actions directly, but retrieves roles from the `roles` directory.

6.2.1 Raspbian compatibility

Some tasks, and in particular those related to lightdm, should not be executed when the base distribution is not fuss- client (or a normal Debian), but Raspbian, which requires some specific customizations; for these the condition `when: ansible_lsb.id != "Raspbian"` is used.

6.3 Debian packages

The repository involves the generation of two .deb packages, `fuss-client` and `fuss-client-dependencies`; the former contains the actual fuss-client, while the latter is a metapackage that depends on all the packages installed by the ansible playbook.

`fuss-client-dependencies` is not required for the use of fuss-client, but is added for convenience to pre-install (and especially pre-download) all necessary packages.

6.3.1 Version numbers

The `fuss-client` package is native, so the version number is of the type X.Y.Z where X is the corresponding debian version number (e.g. 8 for jessie, 9 for stretch, 10 for buster).

6.4 HOWTO

6.4.1 Scripts at startup

To install scripts on fuss-client that run at startup, the recommended method is to use systemd units.

To do this, install the desired script in `/usr/local/bin` (or `sbin`, if it makes sense for it to be run only by `root`), for example as `/usr/local/bin/my_script.sh` with run permissions, then create the file `/etc/systemd/system/my-script.service` with the following contents:

```
[Unit]
Description=My script doing things
After=network.target

[Service]
ExecStart=/usr/local/bin/my_script.sh

[Install]
WantedBy=multi-user.target
```

And enable the unit.

In ansible, you will need tasks like the following:

```
- name: Script to do
  things copy:
      dest:
      /usr/local/bin/my_script.sh src:
      my_script.sh
      mode: 0755
- name: Do things at
  startup copy:
      dest: /etc/systemd/system/my-
      script.service src: my-script.service
- name: Enable doing things at
  startup systemd:
```

(continues on next page)

(continued from previous page)

```
    enabled: yes
    name: do-things
```

CHAPTER 7

Cloud-init image creation

Index

Starting with FUSS 10 (Debian Buster), full images per server virtual machine will be generated in the dump format of the Proxmox platform adopted by the project, so as to simplify the initial installation and configuration of a FUSS server.

The images rely on `cloud-init in` a way that allows network management directly from the web interface, once you have imported them and associated a volume for `cloud-init with` them (details are explained in the fuss-tech-guide.

7.1 Virtual machine creation

Create a new virtual machine (VM) on Proxmox, at the initial stage the single network interface created by the wizard is sufficient, use the bridge associated with the external network.

On the virtual machine you have to do a routine installation of Debian with *netinstall* (get yourself the latest version of the ISO and load it inside `/var/lib/vz/template/iso`):

```
# cd cd /var/lib/vz/template/iso/
# wget https://cdimage.debian.org/cdimage/bookworm_di_rc4/amd64/iso-cd/debian-
↪bookworm-DI-rc4-amd64-netinst.iso
```

In the creation use the previous image as the CDROM and otherwise use the default values for everything, including the 32 G disk size, taking care, however, to have the latter installed on the storage `local` in *qcow* format (which possibly allows you to directly distribute the file you will get as a disk image for use by other platforms).

Once the wizard is complete, add the second interface for the internal network, and the possible third for the captive portal, and make sure that both these and the first interface are unchecked for firewall use.

Also, edit the record and activate the Discard checkmark.

7.2 Install Debian

Carry out the installation in the ordinary way; you can also use any DHCP for the network, the configura- tion you choose during installation will still be overwritten by the one you will set during deployment with `cloud-init`.

The specific steps that need to be taken during installation are:

- you set the default password for `root: fuss`
- you use a normal user: `fuss` (or other, it will still be deleted)
- you configure the network in the easiest way to access on the machine
- Choose manual disk partitioning,
  - create a first primary partition of 4G as swap
  - Create a second primary partition with rest of disk as root
- on the *Software Selection* screen choose only *SSH Server* and *Standard System Utilities* (the rest will install later).

7.3 Preparation of basic installation

Once the basic installation is complete, you can move on to preparing the server. Since the default installation blocks SSH access to root, either use the temporary normal user created during installation and use `su or sudo`, or log onto the console via the web. All of the following operations are to be performed by the `root` user.

The first step is to enable SSH access to `root` right away by entering `/etc/ssh/sshd_config.d/root.conf` inside:

```
# enable root password access
PermitRootLogin yes
```

and restart `sshd` with `service ssh restart`; it is not necessary to configure the use of keys in `authorized_keys` at this stage, these can be set at any later time via `cloud-init`.

Then remove the user created during installation with:

```
# userdel -r fuss
```

You will first need to install `cloud-init` and some programs:

- `gnupg` to be able to import APT keys from FUSS;
- `resolvconf` to be able to manage DNS configuration through `cloud-init`;
- `bind9` in order to immediately configure the machine in a mode compatible with the configuration that will be set by `fuss-server`
- `quotatool` to enable quotas in case you add a separate home disk;

```
# apt install cloud-init gnupg resolvconf bind9 quotedol
```

To avoid the resolution problems with IPv6 addresses that have occurred in some cases, Bind should be configured from the start to use IPv4 only, so edit in `/etc/default/named` the last line in:

```
OPTIONS="-4 -u bind"
```

And restart Bind with `systemctl restart named`.

You must then properly configure `/etc/apt/sources.list` to use fuss repositories, use content similar to the following:

```
# bookworm sources
deb http://deb.debian.org/debian/ bookworm main
deb-src http://deb.debian.org/debian/ bookworm main

deb http://security.debian.org/debian-security bookworm-security main
deb-src http://security.debian.org/debian-security bookworm-security main

# fuss-sources
deb [signed-by=/etc/apt/keyrings/fuss-keyring.gpg] http://archive.fuss.bz.it/ ␣
↪bookworm main
#deb [signed-by=/etc/apt/keyrings/fuss-keyring.gpg] http://archive.fuss.bz.it/ ␣
↪bookworm-proposed-updates main
```

(usually just add the last two lines), also you have to import the package signing GPG key, with:

```
# wget -qO /etc/apt/keyrings/fuss-keyring.gpg https://archive.fuss.bz.it/apt.gpg
```

With the switch to Debian Buster and following by default the network interfaces take on the new hardware-dependent names, so instead of `eth0` and `eth1` you will have names like `ens18` or `ens19`. However, configuring them with `cloud-init` automatically brings the traditional names back into use, so there is no need to worry about this change in configurations, with one exception: the possible third interface dedicated to the captive portal, which does not need to be configured.

Since this is not configured it will take the default name, varying from what you would have with a Debian Jessie. However, you can avoid the change (allowing the unchanged reuse of the names used in a previous `fuss-server.yaml`) by appropriately configuring the kernel boot options by entering in `/etc/default/grub` the line:

```
GRUB_CMDLINE_LINUX="net.ifnames=0 biosdevname=0"
```

and running `update-grub`; this will still call the interfaces `eth0`, `eth1` and `eth2`.

Finally, to prevent the configuration entry for `lo` present in `/etc/network/interfaces` from overwriting the same entry that `cloud-init` configures with its own file under `/etc/network/interfaces.d` (`50-cloud-init.cfg`) move the line that includes the configurations from this directory to the bottom of the file, basically `/etc/network/interfaces` will have to be something like:

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
car lo
iface lo inet loopback

source /etc/network/interfaces.d/*
```

Note: Keep in mind that with this /etc/network/interfaces will not automatically configure any network interfaces, and if you do not configure the same via `cloud-init` the machine will be reachable

only on the console, so if you need to access it via the network to work on it before creating the `cloud-init` image leave the interface configuration you intend to use present and remove it in the cleanup step.

Note: Keep in mind that with this configuration you manage the contents of `/etc/resolv.conf` with `cloud-init`, but this file is also written directly by the `fuss-server` command, so take care to set via `cloud-init` the same domain name that you will then set in `fuss-server` and always indicate as DNS server `127.0.0.1`.

7.4 `Cloud-init` configuration

The default configuration installed by `cloud-init` is to create a default user and block access as `root`. The design choice is to provide direct keyed `root` access, with the option of using a password, so some changes must be made in `/etc/cloud/cloud.cfg`. First, the two lines related to the management of local users created by default must be changed, as in the following example:

```
# A set of users which may be applied and/or used by various modules.
# when a 'default' entry is found it will reference the 'default_user'
# from the distro configuration specified below
users:
   - name: root
```

and then you need to prevent the configuration that disables the use of the `root` user from being applied, by configuring
`disable_root` to `false` as in:

```
# If this is set, 'root' will not be able to ssh in and they
# will get a message to login instead as the above $user (debian)
disable_root: false
```

finally then add to the end of the file the additional configuration that allows `cloud-init` to manage `/etc/hosts`:

```
manage_etc_hosts: true
```

Next, add under `/etc/cloud/cloud.cfg.d/` the file `10_fuss.cfg`, with the following contents (note: spacing should **not** contain tabs):

```
apt:
    preserve_sources_list: true
packages:
  - fuss-server
```

7.5 Cleaning

Once the previous configurations have been made, proceed to clean the image of all spurious data, disconnect, and reconnect by running the command:

```
# set +o history
```

To disable the history.

If you left the main network interface configuration used during installation inside `/etc/network/interfaces` remove it.

Then run (as root) the commands:

```
> .bash_history
apt clean
find /etc -name "*~" -delete
journalctl --rotate
journalctl --vacuum-time=1s
fstrim /
cd /var/log/
> syslog
> auth.log
> cloud-init.log
> cloud-init-output.log
> debugging
> dpkg.log
> messages
> kern.log
> user.log
> daemon.log
> installer/syslog
> wtmp
> btmp
```

to do a final cleanup, including recreating empty files modified by the system while it was in use.

7.6 Image generation

Once you have made the previous configurations, stop the virtual machine and run a backup of it, asking for compression (with the default values in zst format). You will find the backup file under `/var/lib/vz/dump` (or in the directory you configured as backup storage) in the form:

```
vzdump-qemu-VMID-ANNO_ME_GI-OR_MI_SE.vma.zst
```

7.7 Publication

After any tests, upload the file to the machine where it will be published, such as to the root user's home with the command:

```
$ scp /var/lib/vz/dump/vzdump-qemu-106-2023_06_05-16_06_42.vma.zst \
      root@iso.fuss.bz.it:
```

then connect to the machine itself (`ssh root@iso.fuss.bz.it`) and run the command:

```
# cd /root
# ./release_cloud_image.sh vzdump-qemu-106-2023_06_05-16_06_42.vma.zst
```

which takes care of moving the file to the correct destination and generating checksums and related signatures.

Then update the file `/var/www/iso/cloud-init/changelog.txt` with the list of changes in the current release.

The image is now available for download at https://iso.fuss.bz.it/cloud-init/

CHAPTER 8

ISO FUSS 9

In FUSS 9 (Debian stretch) live ISOs (complete with text and graphical installer) were generated using `live-wrapper`.

The versions present in Debian buster, that is, as of August 2018, are used:

```
live-wrapper (0.7)
vmdebootstrap (1.11-1)
```

8.1 Build

8.1.1 Setup

- On a Debian buster installation or later, install live-wrapper:

```
# apt install live-wrapper
```

- Copy the file `/usr/share/live-wrapper/customise.sh`:

```
# cp /usr/share/live-wrapper/customise.sh fuss-customise.sh
```

below the line:

```
. /usr/share/vmdebootstrap/common/customise.lib
```

add:

```
# overriden from the above for FUSS
prepare_apt_source() {}
    # handle the apt source
    mv ${rootdir}/etc/apt/sources.list.d/base.list ${rootdir}/etc/apt/
    echo "deb $1 $2 main contrib non-free" > ${rootdir}/etc/apt/sources.list
    echo "deb-src $1 $2 main contrib non-free" >> ${rootdir}/etc/apt/sources.
↪list
    echo "deb http://archive.fuss.bz.it/ stretch main" >> ${rootdir}/etc/apt/
↪sources.list
    wget -qO ${rootdir}/tmp/fuss-apt.key https://archive.fuss.bz.it/apt.key
```

(continues on next page)

(continued from previous page)

```
    chroot ${rootdir} apt-key add /tmp/fuss-apt.key
    chroot ${rootdir} apt -qq -y update > /dev/null 2>&1
}
```

8.1.2 Build

To generate the ISO of FUSS 9 for amd64 architecture, run the following command:

```
lwr -o fuss9-live-amd64.iso -d stretch --architecture=amd64 --customise=./fuss-.
↪customise.sh -m http://ftp.de.debian.org/debian/ -e "fuss-client fuss-kids fuss-
↪children fuss-education fuss-graphics fuss-language-support fuss-multimedia fuss-
↪extra-multimedia fuss-net fuss-office fuss-various"
```

To generate the ISO of FUSS 9 for i386 architecture, run the following command:

```
lwr -o fuss9-live-i386.iso -d stretch --architecture=i386 --customise=./fuss-.
↪customise.sh -m http://ftp.de.debian.org/debian/ -e "fuss-client fuss-kids fuss-
↪children fuss-education fuss-graphics fuss-language-support fuss-multimedia fuss-
↪extra-multimedia fuss-net fuss-office fuss-various"
```

8.2 See also.

- https://live-wrapper.readthedocs.io/en/latest/
- https://wiki.debian.org/vmdebootstrap

CHAPTER 9

ISO FUSS 10

With FUSS 10, the tool adopted to customize an existing ISO image is called `remaster-iso` (https://github.com/unixabg/remaster-iso).

9.1 Creating the live ISO of FUSS

`remaster-iso` is only available as a package starting with Debian 11 "bullseye". If you are working with Debian 10 you must, peretheless, clone the `remaster-iso` project from GitHub and move to the created folder:

```
git clone https://github.com/unixabg/remaster-iso.git
cd remaster-iso
```

The following shows how to customize the Debian Live amd64 Xfce image found at https://cdimage.debian.org/debian-cd/current-live/amd64/iso-hybrid/debian-live-10.10.0-amd64-xfce.iso with

```
wget https://cdimage.debian.org/debian-cd/current-live/amd64/iso-hybrid/debian-
↪live-10.10.0-amd64-xfce.iso
```

Should non-free firmware also be needed, the image from which to start is as follows: https://cdimage.debian.org/images/unofficial/non-free/images-including-firmware/current-live/amd64/iso-hybrid/debian-live-10.10.0-amd64-xfce+nonfree.iso and the command to give to download it is:

```
wget https://cdimage.debian.org/images/unofficial/non-free/images-including-
↪firmware/current-live/amd64/iso-hybrid/debian-live-10.10.0-amd64-xfce+nonfree.iso
```

Edit the `remaster-iso.conf` configuration file. Below is an example:

```
######################
## remaster settings
######################
_BASEDIR=$(pwd)
_ISOExtractPath="${_BASEDIR}/iso-extract"
_ISOLivePath="live"
_ISOMountPoint="${_BASEDIR}/iso-mount"
_ISOName=""
_ISOTargetName="fuss-10-amd64-live-light"
_ISOTargetTitle="FUSS 10 amd64 live light"
```

(continues on next page)

(continued from previous page)

```
_VER="0.9.4"
```

Extract the .iso file

```
./remaster-extract -i debian-live-10.10.0-amd64-xfce.iso
```

Then run the `remaster-squashfs-editor` command and select "C" by pressing ENTER:

```
./remaster-squashfs-editor
```

```
################################
    remaster-squashfs-editor
    remaster-iso version
    0.9.3
################################
  (C)hroot - Chroot in to the filesystem.squashfs + psu-*.squashfs stack.
  (J)oin   - Join the partial squashfs update files to new single psu-
  DATE.squashfs (N)ew      - New master filesystem.squashfs file which joins all
  *.squashfs files
↪to new single filesystem.squashfs
  E(x)it   - Exit the program with no
action Enter choice [C , J , N , X] C
```

Edit the repository file if necessary

```
nano /etc/apt/sources.list
```

```
# ##### Debian Main Repos
deb http://ftp.it.debian.org/debian/ buster main
deb-src http://ftp.it.debian.org/debian/ buster main

deb http://ftp.it.debian.org/debian/ buster-updates main
deb-src http://ftp.it.debian.org/debian/ buster-updates main

deb http://security.debian.org/debian-security buster/updates main
deb-src http://security.debian.org/debian-security buster/updates main

deb http://ftp.debian.org/debian buster-backports main
deb-src http://ftp.debian.org/debian buster-backports main

# ##### FUSS Main Repo
deb http://archive.fuss.bz.it/ buster main
deb-src http://archive.fuss.bz.it/ buster main
```

If contrib and non-free packages are also needed, the file must be

```
# ##### Debian Main Repos
deb http://ftp.it.debian.org/debian/ buster main contrib non-free
deb-src http://ftp.it.debian.org/debian/ buster main contrib non-free

deb http://ftp.it.debian.org/debian/ buster-updates main contrib non-free
deb-src http://ftp.it.debian.org/debian/ buster-updates main contrib non-free

deb http://security.debian.org/debian-security buster/updates main contrib non-
free deb-src http://security.debian.org/debian-security buster/updates main
contrib non-
↪free

deb http://ftp.debian.org/debian buster-backports main contrib non-free
deb-src http://ftp.debian.org/debian buster-backports main contrib non-free

# ##### FUSS Main Repo
```

(continues on next page)

(continued from previous page)

```
deb http://archive.fuss.bz.it/ buster main contrib non-free
deb-src http://archive.fuss.bz.it/ buster main contrib non-free
```

Update packages

```
apt update
apt install curl wget apt-transport-https dirmngr
wget -qO - https://archive.fuss.bz.it/apt.key | sudo apt-key add -
apt update && apt full-upgrade
```

Install `fuss-client`

```
apt install fuss-client
```

Run the command to configure FUSS standalone (desktop)

```
fuss-client --iso --standalone [--light] [--unofficial] [--local=LOCAL] --domain␣
↪fuss.bz.it.
```

where

- `--light` keeps the image light without installing educational metapackages;
- `--unofficial` allows you to install non-free debian firmware;
- `--local=LOCAL` allows you to choose the default language, where `LOCAL` is, by way of example, in the form de_DE.UTF-8.

Download the repository key `archive.fuss.bz.it` that will be used by Calamares (https://calamares. io/) during installation:

```
curl https://archive.fuss.bz.it/apt.key | gpg --dearmor > /usr/share/keyrings/fuss-.
↪keyring.gpg
```

Edit the `/usr/sbin/sources-final` script that will write the repositories during installation:

```
#!/bin/sh
# Writes the final sources.list
files.

CHROOT=$(mount | grep proc | grep calamares | awk '{print $3}' | sed -e "s#/proc##g
↪")
RELEASE="buster"
FUSS_KEY="/usr/share/keyrings/fuss-keyring.gpg"

cat << EOF > $CHROOT/etc/apt/sources.list
# See https://wiki.debian.org/SourcesList for more information.
deb http://deb.debian.org/debian $RELEASE main
deb-src http://deb.debian.org/debian $RELEASE main

deb http://deb.debian.org/debian $RELEASE-updates main
deb-src http://deb.debian.org/debian $RELEASE-updates main

deb http://security.debian.org/debian-security/ $RELEASE/updates main
deb-src http://security.debian.org/debian-security/ $RELEASE/updates main
EOF

cat << EOF > $CHROOT/etc/apt/sources.list.d/deb_debian_org_debian.list
deb http://deb.debian.org/debian $RELEASE-backports main
deb-src http://deb.debian.org/debian $RELEASE-backports main
EOF
```

(continues on next page)

(continued from previous page)

```
cat << EOF > $CHROOT/etc/apt/sources.list.d/archive_fuss_bz_en.list
deb [signed-by=/usr/share/keyrings/fuss-keyring.gpg] http://archive.fuss.bz.it/
↪$RELEASE main contrib.
deb-src [signed-by=/usr/share/keyrings/fuss-keyring.gpg] http://archive.fuss.bz.it/
↪$RELEASE main contrib
EOF

if [ -f $FUSS_KEY ] ; then
    cp $FUSS_KEY $CHROOT/usr/share/keyrings/fuss-keyring.gpg
fi

exit 0
```

For unofficial images the file must be

```
#!/bin/sh
# Writes the final sources.list
files.

CHROOT=$(mount | grep proc | grep calamares | awk '{print $3}' | sed -e "s#/proc##g
↪")
RELEASE="buster"
FUSS_KEY="/usr/share/keyrings/fuss-keyring.gpg"

cat << EOF > $CHROOT/etc/apt/sources.list
# See https://wiki.debian.org/SourcesList for more information.
deb http://deb.debian.org/debian $RELEASE main contrib non-free
deb-src http://deb.debian.org/debian $RELEASE main contrib non-free

deb http://deb.debian.org/debian $RELEASE-updates main contrib non-free
deb-src http://deb.debian.org/debian $RELEASE-updates main contrib non-free

deb http://security.debian.org/debian-security/ $RELEASE/updates main contrib non-.
↪free
deb-src http://security.debian.org/debian-security/ $RELEASE/updates main contrib.
↪non-free
EOF

cat << EOF > $CHROOT/etc/apt/sources.list.d/deb_debian_org_debian.list
deb http://deb.debian.org/debian $RELEASE-backports main contrib non-
free
deb-src http://deb.debian.org/debian $RELEASE-backports main contrib non-free
EOF

cat << EOF > $CHROOT/etc/apt/sources.list.d/archive_fuss_bz_en.list
deb [signed-by=/usr/share/keyrings/fuss-keyring.gpg] http://archive.fuss.bz.it/
↪$RELEASE main contrib non-free
deb-src [signed-by=/usr/share/keyrings/fuss-keyring.gpg] http://archive.fuss.bz.it/
↪$RELEASE main contrib non-free EOF

if [ -f $FUSS_KEY ] ; then
    cp $FUSS_KEY $CHROOT/usr/share/keyrings/fuss-keyring.gpg
fi

exit 0
```

Remove unused packages and clear the package cache

```
apt-get
autoremove apt-
get clean
```

Configure live

```
nano /etc/live/config.conf.d/fuss.conf
```

```
LIVE_HOSTNAME="fuss"
LIVE_USERNAME="user"
LIVE_USER_FULLNAME="FUSS Live user"
LIVE_LOCALES="en_US.UTF-8,en_IT.UTF-8,de_AT.UTF-8"
LIVE_TIMEZONE="Europe/Rome"
LIVE_KEYBOARD_LAYOUTS="en,de"
```

Change the default hostname

```
nano /etc/hostname
```

```
fuss
```

To take the panel settings as provided by FUSS, edit a line in the `/lib/live/ config/1170-xfce4-panel` script:

```
nano /lib/live/config/1170-xfce4-panel
```

```
sudo -u "${LIVE_USERNAME}" cp /etc/xdg/xfce4/xfconf/xfce-perchannel-xml/xfce4-.
↪panel.xml /home/"${LIVE_USERNAME}"/.config/xfce4/xfconf/xfce-perchannel-xml/
↪xfce4-panel.xml
```

Should you need it, in the `/lib/live/config` folder are all the scripts called by live for the di- verse configurations. As documentation there is the `live-config` man page where everything is pretty well documented.

Exit and save the changes made in chroot by typing Y and enter

```
root@jarvis:~# exit
Exited the chroot so time to clean up.
Restore original overlay/etc/hosts.
Restore overlay/etc/resolv.conf.
Remove
overlay/root/.bash_history.
##############################################################
 (Y)es save my chroot modifications.
 (N)o do not save my chroot modifications.
Select to save your chroot modifications (default is N):

Y
Now making the updated squashfs 20200614-013407.
Parallel mksquashfs: Using 8 processors
Creating 4.0 filesystem on psu-20200614-013407.squashfs, block size 131072.
```

Launch `./remaster-squashfs-editor` again and choose option `N` then confirm the creation of `filesystem.squashfs` with `Y`:

```
./remaster-squashfs-editor
```

```
###################################
    remaster-squashfs-editor
    remaster-iso version
    0.9.3
###################################
  (C)hroot - Chroot in to the filesystem.squashfs + psu-*.squashfs stack
```

(continues on next page)

(continued from previous page)

```
  (J)      oin- Join the partial squashfs update files to new single psu-
  DATE.squashfs (N)ew-    New master filesystem.squashfs file which joins all
  *.squashfs files
↪to new single filesystem.squashfs
  E(x)     it- Exit the program with no
action Enter choice [C , J , N , X] N
I: New option selected!
I: change directory to target live folder
I: strting mount list and points
operations
I: found ./psu-20200614-020636.squashfs ... setting up mount point of psu-20200614-
↪020636_squashfs
I: mounting ./psu-20200614-020636.squashfs on psu-20200614-020636_squashfs
./psu-20200614-020636_squashfs:./filesystem_squashfs
./psu-20200614-020636_squashfs ./filesystem_squashfs ./psu_overlay
./psu-20200614-020636_squashfs:./filesystem_squashfs
############################################################
 (Y)es, create a new single filesystem.squashfs.
 (N)o, do not create a new single filesystem.squashfs.
Select to create a new single filesystem.squashfs (default is N):

Y
```

Remove the `./iso-extract/live/psu-OOS\*` folder.

```
rm -fr ./iso-extract/live/psu-OOS*.
```

Copy the kernel-related files in the squashfs to the `./iso-extract/live` folder

To do this, launch `./remaster-squashfs-editor` again, choosing the option `(C)hroot`

```
config-4.19.0-16-amd64
config-5.10.0-0.bpo.3-amd64
initrd.img-4.19.0-16-amd64
initrd.img-5.10.0-0.bpo.3-amd64
System.map-4.19.0-16-amd64
System.map-5.10.0-0.bpo.3-amd64
vmlinuz-4.19.0-16-amd64
vmlinuz-5.10.0-0.bpo.3-amd64
```

Exit the chroot environment without making any changes Edit

the files to customize the boot menu to your liking:

```
isolinux/menu.cfg
boot/grub/grub.cfg
```

"The time has come to generate the new ISO.

Verify that the xorriso command in the remaster-compose script has the following parameters:

```
xorriso -as mkisofs -r -D -V "${ISOTargetTitle}" -cache-inodes -J -l -iso-level 3 -
↪isohybrid-mbr /usr/lib/ISOLINUX/isohdpfx.bin -c isolinux/boot.cat -b isolinux/
↪isolinux.bin -no-emul-boot -boot-load-size 4 -boot-info-table -eltorito-alt-.
↪boot -e boot/grub/efi.img -no-emul-boot -isohybrid-gpt-basdat -o "${BASEDIR}/$
↪↪{BUILDDATE}-${ISOTargetName}.iso" .
```

Then finish with the `./remaster-compose` command to generate the .iso file

```
./remaster-compose
```

At the end of the script you will find the new .iso image in your current working folder.

Note: For subsequent upgrades and customizations, it will be sufficient to start from the ISO image created precedently by making only the necessary changes and using the three `remaster-iso` scripts as indicated above.

CHAPTER 10

ISO FUSS 11

With FUSS 11, the tool adopted to customize an ISO image remains `remaster-iso` (https://github.com/unixabg/remaster-iso).

10.1 Creating the live ISO of FUSS

`remaster-iso` is only available as a package starting with Debian 11 "bullseye". If you are working with Debian 10 you must, peretheless, clone the `remaster-iso` project from GitHub and move to the created folder:

```
git clone https://github.com/unixabg/remaster-iso.git
cd remaster-iso
```

The following shows how to customize the Debian Live amd64 Xfce image found at https://cdimage.debian.org/debian-cd/current-live/amd64/iso-hybrid/debian-live-11.4.0-amd64-xfce.iso with

```
wget https://cdimage.debian.org/debian-cd/current-live/amd64/iso-hybrid/debian-
↪live-11.4.0-amd64-xfce.iso
```

Should non-free firmware also be needed, the image from which to start is as follows: https://cdimage.debian.org/images/unofficial/non-free/images-including-firmware/current-live/amd64/iso-hybrid/debian-live-11.4.0-amd64-xfce+nonfree.iso and the command to give to download it is:

```
wget https://cdimage.debian.org/images/unofficial/non-free/images-including-
↪firmware/current-live/amd64/iso-hybrid/debian-live-11.4.0-amd64-xfce+nonfree.iso
```

Edit the `remaster-iso.conf` configuration file. Below is an example:

```
######################
## remaster settings
######################
_BASEDIR=$(pwd)
_ISOExtractPath="${_BASEDIR}/iso-extract"
_ISOLivePath="live"
_ISOMountPoint="${_BASEDIR}/iso-mount"
_ISOName=""
_ISOTargetName="fuss-11-amd64-live-lite"
_ISOTargetTitle="FUSS 11 amd64 live lite"
```

(continues on next page)

(continued from previous page)

```
_VER="0.9.4"
```

Extract the .iso file

```
./remaster-extract -i debian-live-11.4.0-amd64-xfce.iso
```

Then run the `remaster-squashfs-editor` command and select "C" by pressing ENTER:

```
./remaster-squashfs-editor
```

```
################################
    remaster-squashfs-editor
    remaster-iso version
    0.9.3
################################
  (C)hroot - Chroot in to the filesystem.squashfs + psu-*.squashfs stack.
  (J)oin   - Join the partial squashfs update files to new single psu-
  DATE.squashfs (N)ew      - New master filesystem.squashfs file which joins all
  *.squashfs files
↪to new single filesystem.squashfs
  E(x)it   - Exit the program with no
action Enter choice [C , J , N , X] C
```

Edit the repository file if necessary

```
nano /etc/apt/sources.list
```

```
# ##### Debian Main Repos
deb http://deb.debian.org/debian bullseye main
deb-src http://deb.debian.org/debian bullseye main

deb http://deb.debian.org/debian bullseye-updates main
deb-src http://deb.debian.org/debian bullseye-updates main

deb http://deb.debian.org/debian-security/ bullseye-security main
deb-src http://deb.debian.org/debian-security/ bullseye-security main
```

```
nano /etc/apt/sources.list.d/deb_debian_org_debian.list
```

```
# ##### Debian Backports Repo
deb http://deb.debian.org/debian bullseye-backports main
deb-src http://deb.debian.org/debian bullseye-backports main
```

```
nano /etc/apt/sources.list.d/archive_fuss_bz_en.list
```

```
# ##### FUSS Main Repo
deb [signed-by=/usr/share/keyrings/fuss-keyring.gpg] http://archive.fuss.bz.it/
↪bullseye main
deb-src [signed-by=/usr/share/keyrings/fuss-keyring.gpg] http://archive.fuss.bz.it/
↪bullseye main
```

```
# ##### Debian Main Repos
deb http://deb.debian.org/debian bullseye main contrib non-free
deb-src http://deb.debian.org/debian bullseye main contrib non-free
```

If contrib and non-free packages are also needed, the file should be modified as follows:

```
nano /etc/apt/sources.list
```

(continues on next page)

(continued from previous page)

```
deb http://deb.debian.org/debian bullseye-updates main contrib non-free
deb-src http://deb.debian.org/debian bullseye-updates main contrib non-free

deb http://deb.debian.org/debian-security/ bullseye-security main contrib non-
free deb-src http://deb.debian.org/debian-security/ bullseye-security main
contrib non-
↪free
```

```
nano /etc/apt/sources.list.d/deb_debian_org_debian.list
```

```
# ##### Debian Backports Repo
deb http://deb.debian.org/debian bullseye-backports main contrib non-free
deb-src http://deb.debian.org/debian bullseye-backports main contrib non-free
```

```
nano /etc/apt/sources.list.d/archive_fuss_bz_en.list
```

```
# ##### FUSS Main Repo
deb [signed-by=/usr/share/keyrings/fuss-keyring.gpg] http://archive.fuss.bz.it/ ␣
↪bullseye main contrib non-free
deb-src [signed-by=/usr/share/keyrings/fuss-keyring.gpg] http://archive.fuss.bz.it/
↪bullseye main contrib non-free
```

Update packages

```
apt update
apt install curl wget apt-transport-https dirmngr
wget -qO - https://archive.fuss.bz.it/apt.key | gpg --dearmour > /usr/share/
↪keyrings/fuss-keyring.gpg
apt update && apt full-upgrade
```

Install `fuss-client`

```
apt install fuss-client
```

Run the command to configure FUSS standalone (desktop)

```
fuss-client --iso --standalone --light [--unofficial] [--local=LOCAL] --domain ␣
↪fuss.bz.it.
```

where

- `--light` keeps the image light without installing educational metapackages;
- `--unofficial` allows you to install non-free debian firmware;
- `--local=LOCAL` allows you to choose the default language, where `LOCAL` is, by way of example, in the form de_DE.UTF-8.

Specifically:

- For **ISO FUSS official lite**:

```
fuss-client --iso --standalone --light ---domain fuss.bz.it
```

- For the **full official ISO FUSS**:

```
fuss-client --iso --standalone --domain fuss.bz.it
```

- For the **ISO FUSS "unofficial" light**:

```
fuss-client --iso --standalone --light --unofficial --domain fuss.bz.it
```

- For the **full "unofficial" ISO**:

```
fuss-client --iso --standalone --unofficial ---domain fuss.bz.it
```

If not done previously, download the repository key `archive.fuss.bz.it` that will be used by Calamares (https://calamares.io/) during installation:

```
curl https://archive.fuss.bz.it/apt.key | gpg --dearmor > /usr/share/keyrings/fuss-.
↪keyring.gpg
```

Edit the `/usr/sbin/sources-final` script that will write the repositories during installation:

```
#!/bin/sh
# Writes the final sources.list
files.

CHROOT=$(mount | grep proc | grep calamares | awk '{print $3}' | sed -e "s#/proc##g
↪")
RELEASE="bullseye"
FUSS_KEY="/usr/share/keyrings/fuss-keyring.gpg"

cat << EOF > $CHROOT/etc/apt/sources.list
# See https://wiki.debian.org/SourcesList for more information.
deb http://deb.debian.org/debian $RELEASE main
deb-src http://deb.debian.org/debian $RELEASE main

deb http://deb.debian.org/debian $RELEASE-updates main
deb-src http://deb.debian.org/debian $RELEASE-updates main

deb http://security.debian.org/debian-security/ $RELEASE-security main
deb-src http://security.debian.org/debian-security/ $RELEASE-security main
EOF

cat << EOF > $CHROOT/etc/apt/sources.list.d/deb_debian_org_debian.list
deb http://deb.debian.org/debian $RELEASE-backports main
deb-src http://deb.debian.org/debian $RELEASE-backports main
EOF

cat << EOF > $CHROOT/etc/apt/sources.list.d/archive_fuss_bz_en.list
deb [signed-by=/usr/share/keyrings/fuss-keyring.gpg] http://archive.fuss.bz.it/
↪$RELEASE main
deb-src [signed-by=/usr/share/keyrings/fuss-keyring.gpg] http://archive.fuss.bz.it/
↪$RELEASE main
EOF

if [ -f $FUSS_KEY ] ; then
    cp $FUSS_KEY $CHROOT/usr/share/keyrings/fuss-keyring.gpg
fi

exit 0
```

For unofficial images the file must be

```
#!/bin/sh
# Writes the final sources.list
files.

CHROOT=$(mount | grep proc | grep calamares | awk '{print $3}' | sed -e "s#/proc##g
↪")
RELEASE="bullseye"
FUSS_KEY="/usr/share/keyrings/fuss-keyring.gpg"
```

(continues on next page)

(continued from previous page)

```
cat << EOF > $CHROOT/etc/apt/sources.list
# See https://wiki.debian.org/SourcesList for more information.
deb http://deb.debian.org/debian $RELEASE main contrib non-free
deb-src http://deb.debian.org/debian $RELEASE main contrib non-free

deb http://deb.debian.org/debian $RELEASE-updates main contrib non-free
deb-src http://deb.debian.org/debian $RELEASE-updates main contrib non-free

deb http://security.debian.org/debian-security/ $RELEASE-security main contrib non-.
↪free
deb-src http://security.debian.org/debian-security/ $RELEASE-security main contrib.
↪non-free
EOF

cat << EOF > $CHROOT/etc/apt/sources.list.d/deb_debian_org_debian.list
deb http://deb.debian.org/debian $RELEASE-backports main contrib non-
free
deb-src http://deb.debian.org/debian $RELEASE-backports main contrib non-free
EOF

cat << EOF > $CHROOT/etc/apt/sources.list.d/archive_fuss_bz_en.list
deb [signed-by=/usr/share/keyrings/fuss-keyring.gpg] http://archive.fuss.bz.it/
↪$RELEASE main contrib non-free
deb-src [signed-by=/usr/share/keyrings/fuss-keyring.gpg] http://archive.fuss.bz.it/
↪$RELEASE main contrib non-free
EOF

if [ -f $FUSS_KEY ] ; then
    cp $FUSS_KEY $CHROOT/usr/share/keyrings/fuss-keyring.gpg
fi

exit 0
```

Replace the `main.py` file in the `/usr/lib/x86_64-linux-gnu/calamares/modules/displaymanager` folder with the modified version available at the link https://www.fuss.bz.it/utility/calamares/main py.

```
wget -O /usr/lib/x86_64-linux-gnu/calamares/modules/displaymanager/main.py https://
↪www.fuss.bz.it/utility/calamares/main.py
```

Remove unused packages and clear the package cache

```
apt-get
autoremove apt-
get clean
```

Configure live

```
nano /etc/live/config.conf.d/fuss.conf
```

```
LIVE_HOSTNAME="fuss"
LIVE_USERNAME="user"
LIVE_USER_FULLNAME="FUSS Live user"
LIVE_LOCALES="en_US.UTF-8,en_IT.UTF-8,de_AT.UTF-8"
LIVE_TIMEZONE="Europe/Rome"
LIVE_KEYBOARD_LAYOUTS="en,de"
```

Change console encoding

```
nano /etc/default/console-setup
```

by setting

```
CHARMAP="UTF-8"
```

Change the default hostname

```
nano /etc/hostname
```

```
fuss
```

To take the panel settings as provided by FUSS, edit a line in the `/lib/live/ config/1170-xfce4-panel` script:

```
nano /lib/live/config/1170-xfce4-panel
```

```
sudo -u "${LIVE_USERNAME}" cp /etc/xdg/xfce4/xfconf/xfce-perchannel-xml/xfce4-.
↪panel.xml /home/"${LIVE_USERNAME}"/.config/xfce4/xfconf/xfce-perchannel-xml/
↪xfce4-panel.xml
```

Should you need it, in the `/lib/live/config` folder are all the scripts called by live for the di- verse configurations. As documentation there is the `live-config` man page where everything is pretty well documented.

Exit and save the changes made in chroot by typing Y and enter

```
root@jarvis:~# exit
Exited the chroot so time to clean up.
Restore original overlay/etc/hosts.
Restore overlay/etc/resolv.conf.
Remove
overlay/root/.bash_history.
#############################################################
 (Y)es save my chroot modifications.
 (N)o do not save my chroot modifications.
Select to save your chroot modifications (default is N):

Y
Now making the updated squashfs 20200614-013407.
Parallel mksquashfs: Using 8 processors
Creating 4.0 filesystem on psu-20200614-013407.squashfs, block size 131072.
```

Launch `./remaster-squashfs-editor` again and choose option `N` then confirm the creation of `filesystem.squashfs` with `Y`:

```
./remaster-squashfs-editor
```

```
###################################
    remaster-squashfs-editor
    remaster-iso version
    0.9.3
###################################
  (C)hroot - Chroot in to the filesystem.squashfs + psu-*.squashfs stack.
  (J)      oin- Join the partial squashfs update files to new single psu-
  DATE.squashfs (N)ew-    New master filesystem.squashfs file which joins all
  *.squashfs files
↪to new single filesystem.squashfs
  E(x)     it- Exit the program with no
action Enter choice [C , J , N , X] N
I: New option selected!
I: change directory to target live folder
I: strting mount list and points
operations
I: found ./psu-20200614-020636.squashfs ... setting up mount point of psu-20200614-
↪020636_squashfs
I: mounting ./psu-20200614-020636.squashfs on psu-20200614-020636_squashfs
./psu-20200614-020636_squashfs:./filesystem_squashfs
```

(continues on next page)

(continued from previous page)

```
./psu-20200614-020636_squashfs ./filesystem_squashfs ./psu_overlay
./psu-20200614-020636_squashfs:./filesystem_squashfs
#############################################################
 (Y)es, create a new single filesystem.squashfs.
 (N)o, do not create a new single filesystem.squashfs.
Select to create a new single filesystem.squashfs (default is N):

Y
```

Remove the `./iso-extract/live/psu-OOS\*` folder.

```
rm -fr ./iso-extract/live/psu-OOS*.
```

Copy the kernel-related files in the squashfs to the `./iso-extract/live` folder

To do this, launch `./remaster-squashfs-editor` again, choosing the option `(C)hroot`

```
config-4.19.0-16-amd64
config-5.10.0-0.bpo.3-amd64
initrd.img-4.19.0-16-amd64
initrd.img-5.10.0-0.bpo.3-amd64
System.map-4.19.0-16-amd64
System.map-5.10.0-0.bpo.3-amd64
vmlinuz-4.19.0-16-amd64
vmlinuz-5.10.0-0.bpo.3-amd64
```

Exit the chroot environment without making any changes

Let the image `iso-extract/isolinux/splash.png` be modified as needed.

Edit the files to customize the boot menu to your liking. The latest isos produced left only the live option (which includes the Calamares installer) and to remove the `Graphical Debian Installer`, `Debian Installer`, and `Debian Installer with Speech Synthesis` options:

```
iso-extract/isolinux/menu.cfg
iso-extract/boot/grub/grub.cfg
```

It is now time to generate the new ISO by running the command

```
./remaster-compose
```

At the end of the script you will find the new .iso image in your current working folder.

Note: For subsequent upgrades and customizations, it will be sufficient to start from the ISO image created precedently by making only the necessary changes and using the three `remaster-iso` scripts as indicated above.

CHAPTER 11

ISO FUSS 12

With FUSS 12, the tool adopted to customize an ISO image remains `remaster-iso` (https://github.com/unixabg/remaster-iso).

11.1 Creating the live ISO of FUSS

`remaster-iso` is only available as a package starting with Debian 11 "bullseye". If you are working with Debian 10 you must, therefore, clone the `remaster-iso` project from GitHub and move to the created folder:

```
git clone https://github.com/unixabg/remaster-iso.git
cd remaster-iso
```

The following shows how to customize the Debian Live amd64 Xfce image found at https://cdimage.debian.org/debian-cd/current-live/amd64/iso-hybrid/debian-live-11.4.0-amd64-xfce.iso with

```
wget https://cdimage.debian.org/debian-cd/current-live/amd64/iso-hybrid/debian-
↪live-11.4.0-amd64-xfce.iso
```

Should non-free firmware also be needed, the image from which to start is as follows: https://cdimage.debian.org/images/unofficial/non-free/images-including-firmware/current-live/amd64/iso-hybrid/debian-live-11.4.0-amd64-xfce+nonfree.iso and the command to give to download it is:

```
wget https://cdimage.debian.org/images/unofficial/non-free/images-including-
↪firmware/current-live/amd64/iso-hybrid/debian-live-11.4.0-amd64-xfce+nonfree.iso
```

Edit the `remaster-iso.conf` configuration file. Below is an example:

```
######################
## remaster settings
######################
_BASEDIR=$(pwd)
_ISOExtractPath="${_BASEDIR}/iso-extract"
_ISOLivePath="live"
_ISOMountPoint="${_BASEDIR}/iso-mount"
_ISOName=""
_ISOTargetName="fuss-12-amd64-live-lite"
_ISOTargetTitle="FUSS 12 amd64 live lite"
```

(continues on next page)

(continued from previous page)

```
_VER="0.9.4"
```

Extract the .iso file

```
./remaster-extract -i debian-live-12.4.0-amd64-xfce.iso
```

Then run the `remaster-squashfs-editor` command and select "C" by pressing ENTER:

```
./remaster-squashfs-editor
```

```
################################
    remaster-squashfs-editor
    remaster-iso version
    0.9.3
################################
  (C)hroot - Chroot in to the filesystem.squashfs + psu-*.squashfs stack.
  (J)oin   - Join the partial squashfs update files to new single psu-
  DATE.squashfs (N)ew      - New master filesystem.squashfs file which joins all
  *.squashfs files
↪to new single filesystem.squashfs
  E(x)it   - Exit the program with no
action Enter choice [C , J , N , X] C
```

Modify the repository file if necessary

```
nano /etc/apt/sources.list
```

```
# ##### Debian Main Repos
# See https://wiki.debian.org/SourcesList for more information.
deb http://deb.debian.org/debian bookworm main contrib non-free-firmware non-
free deb-src http://deb.debian.org/debian bookworm main contrib non-free-
firmware non-
↪free

deb http://deb.debian.org/debian bookworm-updates main contrib non-free-firmware
↪non-free
deb-src http://deb.debian.org/debian bookworm-updates main contrib non-free-.
↪non-free firmware

deb http://security.debian.org/debian-security/ bookworm-security main contrib non-.
↪free-firmware non-free
deb-src http://security.debian.org/debian-security/ bookworm-security main contrib
↪non-free-firmware non-free
nano /etc/apt/sources.list.d/deb_debian_org_debian.list
```

```
# ##### Debian Backports Repo
deb http://httpredir.debian.org/debian bookworm-backports main contrib non-free-
↪non-free firmware
```

```
nano /etc/apt/sources.list.d/archive_fuss_bz_en.list
```

```
# ##### FUSS Main Repo
deb [signed-by=/usr/share/keyrings/fuss-keyring.gpg] http://archive.fuss.bz.it/
↪bookworm main contrib non-free
deb-src [signed-by=/usr/share/keyrings/fuss-keyring.gpg] http://archive.fuss.bz.it/
↪bookworm main contrib non-free
```

If contrib and non-free packages are also needed, the file should be modified as follows:

```
nano /etc/apt/sources.list
```

```
# ##### Debian Main Repos
# See https://wiki.debian.org/SourcesList for more information.
deb http://deb.debian.org/debian bookworm main
deb-src http://deb.debian.org/debian bookworm main

deb http://deb.debian.org/debian bookworm-updates main
deb-src http://deb.debian.org/debian bookworm-updates main

deb http://security.debian.org/debian-security/ bookworm-security main
deb-src http://security.debian.org/debian-security/ bookworm-security main
```

```
nano /etc/apt/sources.list.d/deb_debian_org_debian.list
```

```
# ##### Debian Backports Repo
deb http://httpredir.debian.org/debian bookworm-backports main
```

```
nano /etc/apt/sources.list.d/archive_fuss_bz_en.list
```

```
# ##### FUSS Main Repo
deb [signed-by=/usr/share/keyrings/fuss-keyring.gpg] http://archive.fuss.bz.it/ ␣
↪bookworm main
deb-src [signed-by=/usr/share/keyrings/fuss-keyring.gpg] http://archive.fuss.bz.it/
↪bookworm main
```

Update packages

```
apt update
apt install curl wget apt-transport-https dirmngr
wget -qO - https://archive.fuss.bz.it/apt.key | gpg --dearmour > /usr/share/
↪keyrings/fuss-keyring.gpg
apt update && apt full-upgrade
```

Install `fuss-client`

```
apt install fuss-client
```

Run the command to configure FUSS standalone (desktop)

```
fuss-client --iso --standalone --light [--unofficial] [--local=LOCAL] --domain ␣
↪fuss.bz.it.
```

where

- `--light` keeps the image light without installing educational metapackages;
- `--unofficial` allows you to install non-free debian firmware;
- `--local=LOCAL` allows you to choose the default language, where `LOCAL` is, by way of example, in the form de_DE.UTF-8.

Specifically:

- For **ISO FUSS official lite**:

```
fuss-client --iso --standalone --light ---domain fuss.bz.it
```

- For the **full official ISO FUSS**:

```
fuss-client --iso --standalone --domain fuss.bz.it
```

- For the **ISO FUSS "unofficial" light**:

```
fuss-client --iso --standalone --light --unofficial --domain fuss.bz.it
```

- For the **full "unofficial" ISO**:

```
fuss-client --iso --standalone --unofficial ---domain fuss.bz.it
```

If not done previously, download the repository key `archive.fuss.bz.it` that will be used by Calamares (https://calamares.io/) during installation:

```
curl https://archive.fuss.bz.it/apt.key | gpg --dearmor > /usr/share/keyrings/fuss-.
↪keyring.gpg
```

Note: If the fuss-client -standalone . . . should fail with the error:

ERROR: Ansible could not initialize the preferred locale: unsupported locale setting

reconfigure locales by choosing a language by running the command:

```
dpkg-reconfigure locales
```

Edit the `/usr/sbin/sources-final` script that will write the repositories during installation:

```
#!/bin/sh
# Writes the final sources.list
files.

CHROOT=$(mount | grep proc | grep calamares | awk '{print $3}' | sed -e "s#/proc##g
↪")
RELEASE="bookworm"
FUSS_KEY="/usr/share/keyrings/fuss-keyring.gpg"

cat << EOF > $CHROOT/etc/apt/sources.list
# See https://wiki.debian.org/SourcesList for more information.
deb http://deb.debian.org/debian $RELEASE main
deb-src http://deb.debian.org/debian $RELEASE main

deb http://deb.debian.org/debian $RELEASE-updates main
deb-src http://deb.debian.org/debian $RELEASE-updates main

deb http://security.debian.org/debian-security/ $RELEASE-security main
deb-src http://security.debian.org/debian-security/ $RELEASE-security main
EOF

cat << EOF > $CHROOT/etc/apt/sources.list.d/deb_debian_org_debian.list
deb http://deb.debian.org/debian $RELEASE-backports main
deb-src http://deb.debian.org/debian $RELEASE-backports main
EOF

cat << EOF > $CHROOT/etc/apt/sources.list.d/archive_fuss_bz_en.list
deb [signed-by=/usr/share/keyrings/fuss-keyring.gpg] http://archive.fuss.bz.it/
↪$RELEASE main
deb-src [signed-by=/usr/share/keyrings/fuss-keyring.gpg] http://archive.fuss.bz.it/
↪$RELEASE main
EOF

if [ -f $FUSS_KEY ] ; then
    cp $FUSS_KEY $CHROOT/usr/share/keyrings/fuss-keyring.gpg
fi
```

(continues on next page)

(continued from previous page)

```
exit 0
```

For unofficial images the file must be

```
#!/bin/sh
# Writes the final sources.list
files.

CHROOT=$(mount | grep proc | grep calamares | awk '{print $3}' | sed -e "s#/proc##g
↪")
RELEASE="bullseye"
FUSS_KEY="/usr/share/keyrings/fuss-keyring.gpg"

cat << EOF > $CHROOT/etc/apt/sources.list
# See https://wiki.debian.org/SourcesList for more information.
deb http://deb.debian.org/debian $RELEASE main contrib non-free-firmware non-free
deb-src http://deb.debian.org/debian $RELEASE main contrib non-free-firmware
non-free.
↪free

deb http://deb.debian.org/debian $RELEASE-updates main contrib non-free-firmware␣
↪non-free
deb-src http://deb.debian.org/debian $RELEASE-updates main contrib non-free-.
↪non-freefirmware

deb http://security.debian.org/debian-security/ $RELEASE-security main contrib non-.
↪free-firmware non-free
deb-src http://security.debian.org/debian-security/ $RELEASE-security main contrib␣
↪non-free-firmware non-free
EOF

cat << EOF > $CHROOT/etc/apt/sources.list.d/deb_debian_org_debian.list
deb http://deb.debian.org/debian $RELEASE-backports main contrib non-
free
deb-src http://deb.debian.org/debian $RELEASE-backports main contrib non-free
EOF

cat << EOF > $CHROOT/etc/apt/sources.list.d/archive_fuss_bz_en.list
deb [signed-by=/usr/share/keyrings/fuss-keyring.gpg] http://archive.fuss.bz.it/
↪$RELEASE main contrib non-free
deb-src [signed-by=/usr/share/keyrings/fuss-keyring.gpg] http://archive.fuss.bz.it/
↪$RELEASE main contrib non-free
EOF

if [ -f $FUSS_KEY ] ; then
    cp $FUSS_KEY $CHROOT/usr/share/keyrings/fuss-keyring.gpg
fi
exit 0
```

Replace the `main.py` file in the `/usr/lib/x86_64-linux-gnu/calamares/modules/displaymanager` folder with the modified version available at the link https://www.fuss.bz.it/utility/calamares/main py.

```
wget -O /usr/lib/x86_64-linux-gnu/calamares/modules/displaymanager/main.py https://
↪www.fuss.bz.it/utility/calamares/main.py
```

Remove unused packages and clear the package cache

```
apt-get
autoremove apt-
get clean
```

Configure live

```
nano /etc/live/config.conf.d/fuss.conf
```

```
LIVE_HOSTNAME="fuss"
LIVE_USERNAME="user"
LIVE_USER_FULLNAME="FUSS Live user"
LIVE_LOCALES="en_US.UTF-8,en_IT.UTF-8,de_AT.UTF-8"
LIVE_TIMEZONE="Europe/Rome"
LIVE_KEYBOARD_LAYOUTS="en,de"
```

Change console encoding

```
nano /etc/default/console-setup
```

by setting

```
CHARMAP="UTF-8"
```

Change the default hostname

```
nano /etc/hostname
```

```
fuss
```

To take the panel settings as provided by FUSS, edit a line in the `/lib/live/ config/1170-xfce4-panel` script:

```
nano /lib/live/config/1170-xfce4-panel
```

```
sudo -u "${LIVE_USERNAME}" cp /etc/xdg/xfce4/xfconf/xfce-perchannel-xml/xfce4-.
↪panel.xml /home/"${LIVE_USERNAME}"/.config/xfce4/xfconf/xfce-perchannel-xml/
↪xfce4-panel.xml
```

Should you need it, in the `/lib/live/config` folder are all the scripts called by live for the di- verse configurations. As documentation there is the `live-config` man page where everything is pretty well documented.

Exit and save the changes made in chroot by typing Y and enter

```
root@jarvis:~# exit
Exited the chroot so time to clean up.
Restore original overlay/etc/hosts.
Restore overlay/etc/resolv.conf.
Remove
overlay/root/.bash_history.
#########################################################
 (Y)es save my chroot modifications.
 (N)o do not save my chroot modifications.
Select to save your chroot modifications (default is N):

Y
Now making the updated squashfs 20200614-013407.
Parallel mksquashfs: Using 8 processors
Creating 4.0 filesystem on psu-20200614-013407.squashfs, block size 131072.
```

Launch `./remaster-squashfs-editor` again and choose option `N` then confirm the creation of `filesystem.squashfs` with `Y`:

```
./remaster-squashfs-editor
```

```
##################################
    remaster-squashfs-editor
    remaster-iso version
    0.9.3
##################################
  (C)hroot - Chroot in to the filesystem.squashfs + psu-*.squashfs stack.
  (J)      oin- Join the partial squashfs update files to new single psu-
  DATE.squashfs (N)ew-    New master filesystem.squashfs file which joins all
  *.squashfs files
↪to new single filesystem.squashfs
  E(x)     it- Exit the program with no
action Enter choice [C , J , N , X] N
I: New option selected!
I: change directory to target live folder
I: strting mount list and points
operations
I: found ./psu-20200614-020636.squashfs ... setting up mount point of psu-20200614-
↪020636_squashfs
I: mounting ./psu-20200614-020636.squashfs on psu-20200614-020636_squashfs
./psu-20200614-020636_squashfs:./filesystem_squashfs
./psu-20200614-020636_squashfs ./filesystem_squashfs ./psu_overlay
./psu-20200614-020636_squashfs:./filesystem_squashfs
##################################################################
 (Y)es, create a new single filesystem.squashfs.
 (N)o, do not create a new single filesystem.squashfs.
Select to create a new single filesystem.squashfs (default is N):

Y
```

Remove the `./iso-extract/live/psu-OOS\*` folder.

```
rm -fr ./iso-extract/live/psu-OOS*.
```

Copy the kernel-related files in the squashfs to the `./iso-extract/live` folder

```
config-4.19.0-16-amd64
config-5.10.0-0.bpo.3-amd64
initrd.img-4.19.0-16-amd64
initrd.img-5.10.0-0.bpo.3-amd64
System.map-4.19.0-16-amd64
System.map-5.10.0-0.bpo.3-amd64
vmlinuz-4.19.0-16-amd64
vmlinuz-5.10.0-0.bpo.3-amd64
```

To do this launch `./remaster-squashfs-editor` again, choosing the `(C)hroot` option Open another terminal and move to iso-extract/live. Copy there the files from iso-extract/live/psu_overlay/boot/, where the folder was mounted, possibly deleting those with the unwanted versions

```
cp /media/iso/fuss-12-amd64-live-full/iso-extract/live/psu_overlay/boot/* .
```

Exit the chroot environment without making any changes

Let the `iso-extract/isolinux/splash.png` image be modified as needed.

Edit the files to customize the boot menu to your liking. You may decide, for example, to keep only the live option (which includes the Calamares installer) and to remove the `Graphical DebianInstaller`, `Debian Installer`, and `Debian Installer with Speech Synthesis` options:

```
iso-extract/isolinux/menu.cfg
iso-extract/boot/grub/grub.cfg
```

Note: In case the boot menu lacks some options, such as localization support to start the live in the desired language, you can copy parts of code from the files of previous distributions.

It is now time to generate the new ISO by running the command

```
./remaster-compose
```

At the end of the script you will find the new .iso image in your current working folder.

Note: For subsequent upgrades and customizations, it will be sufficient to start from the ISO image created precedently by making only the necessary changes and using the three `remaster-iso` scripts as indicated above.

CHAPTER 12

New versions of Debian

This is the procedure to accomplish the upgrade of a version of FUSS based on a new Debian stable release.

12.1 Update procedure

12.1.1 Software Repository

The git repositories containing the FUSS custom software will need to host the new upstream-specific versions.

The current workflow is as follows:

branch `master`: current debian stable distribution

branch `codename_distribution`: contains code against a specific version of Debian, for backporting and other fixes

When a new Debian is released, it is then necessary to create the branch related to the new oldstable, and continue on master for the current stable version.

For example, to switch from stretch to buster, on all repositories that contain software published in the FUSS archive:

```
(master) $ git pull
(master) $ git checkout -b stretch
(stretch) $ git push -u origin stretch
(stretch) $ git checkout master
(master) $ # continue with the changes...
```

12.1.2 Update, build and package testing

Each new package must at a minimum:

- Be up to date with the Debian reference standard for your version (see https://www.debian.org/doc/debian-policy/).

- Have a version (and its changelog) that shows, as the major version number, that of the reference Debian distribution. For example, a package for Debian Buster will have as version 10.x.x.
- Be up-to-date with respect to new versions of the dependencies in the new version of the distribution.
- It will need to be built *and tested* on an installation of the reference Debian version. See

Packages and Repositories for build and subsequent upload instructions.

fuss-server and fuss-client

FUSS Server and *FUSS Client* take action on the configuration files of many packages: specific checks should be made on their operation when updating them.

- For files where sections are inserted in configuration files (task `lineinfile` and `blockinfile`) that the sections are still inserted in the appropriate place.
- For files that are completely overwritten (task `copy` and `template`) it is generally the case to repar- t from the default configuration file of the new debian version and reapply the necessary changes (so that any new default settings are received).

12.1.3 ISO images

TBD

12.2 Information on upstream development

12.2.1 Release Dates

Unlike other distributions, Debian does not set release dates, giving higher priority to the quality of the distribution. However, *freeze* dates are set, which allows one to estimate with approximation of a few months when the next release will occur.

The freeze, occurs every other winter; exact dates are announced well in advance and posted on https://release.debian.org/; releases generally occur[1] during the summers of odd-numbered years.

As of the date of the freeze, no more changes are made to the versions of packages in the distribution, only targeted fixes for sufficiently important bugs are accepted; this for FUSS means that any version upgrade tests will already match the behavior of the released version.

[1] Additional statistics can be found at https://wiki.debian.org/DebianReleases#Release_statistics Note that the date of the freeze was moved forward two months between jessie and stretch.

CHAPTER 13

Virtual machines with libvirt+qemu/kvm for fuss-server and fuss-client

For doing fuss-server and fuss-client testing it is useful to have virtual machines available; especially if you are working on debian stretch or later (where VirtualBox is no longer available) it is convenient to use qemu/kvm via libvirt.

13.1 Installation and configuration

13.1.1 Installation of libvirt

- Install the following packages:

```
apt install qemu libvirt-clients libvirt-daemon virtinst \
libvirt-daemon-system virt-viewer virt-manager dnsmasq-
base
```

- Add your user to the `libvirt` and `kvm` groups:

```
adduser $USER libvirt
adduser $USER kvm
```

Once the user is part of the groups (e.g., after logout/re-login) one can use `virt-manager` to manage virtual machines and networks through a graphical interface similar to that of VirtualBox.

Wanting to use the command line instead, you can continue with this guide.

13.1.2 Network configuration

fuss-server requires the configuration of at least two, in some cases three, network cards: one with Internet access and two on an isolated network.

To create these network interfaces from the command line, one must write their configuration file and pass it to the `virsh net-define` command

For the natted network, create the natted `.xml` file with the following contents, replacing `8.8.8.8` with the address of an appropriate dns server:

```
<network>
  <name>natted</name>
  <forward mode='nat' />
  <bridge name='virbr7' stp='on' delay='0'/>
  <dns>
    <forwarder addr='8.8.8.8'/>
  </dns>
  <ip address='192.168.7.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.7.128' end='192.168.7.254'/>
    </dhcp>
  </ip>
</network>
```

Tip: With this configuration on the host, a `virbr7` network interface will be configured with the address `192.168.7.1` assigned.

Instead, virtual machines will have a dchp that will assign them addresses in the range from `192.168.7.128` to `192.168.7.254`, and DNS requests will be forwarded to server `8.8.8.8`.

then launch, as root:

```
# virsh net-define
natted.xml # virsh net-start
natted
```

Similarly, the following can be used for isolated interfaces in the `isolated.xml` file:

```
<network>
  <name>isolated</name>
  <bridge name='virbr6' stp='on' delay='0'/>
  <ip address='192.168.6.253' netmask='255.255.255.0'>
  </ip>
</network>
```

And in the `isolated2.xml` file:

```
<network>
  <name>isolated2</name>
  <bridge name='virbr8' stp='on' delay='0'/>
  <ip address='192.168.8.253' netmask='255.255.255.0'>
  </ip>
</network>
```

And as before, always with root user:

```
# virsh net-define
isolated.xml # virsh net-start
isolated
# virsh net-define isolated2.xml
# virsh net-start isolated2
```

In this way the interfaces are defined, but they will not be started automatically; use the following commands to do so:

```
# virsh net-autostart natted
# virsh net-autostart isolated
# virsh net-autostart
isolated2
```

or use `net-start` for each interface when you need it.

13.1.3 Virtual machine creation

To create the machine that will host the server, after enabling the network interfaces and downloading the fuss-server iso run the following command:

```
$ virt-install --connect qemu:///system --name fuss_server --memory 1024 \
  --cdrom $PATH_ISO_FUSS-SERVER --network network=natted \.
  --network network=isolated --network network=isolated2 \
  --disk size=16,format=qcow2 --os-variant debian8
```

this will create a virtual machine that will boot from the installer iso and open a virt-viewer window to check it. At the end of the installation you can login and proceed with the "Installation of Fuss Server

Once the server is installed and configured you can do the same thing for one (or more) client machines:

```
$ virt-install --connect qemu:///system --name fuss_client --memory 1024 \
  --cdrom $PATH_ISO_FUSS-CLIENT --network network=isolated \.
  --disk size=24,format=qcow2 --os-variant debian9
```

13.2 Management

13.2.1 Machine Booting

In order to start subsequent times of machines, it is necessary:

- If the host has been turned off, and auto-start of network interfaces has not been configured, enable them:

```
# virsh net-start natted
# virsh net-start isolated
# virsh net-start
isolated2
```

- Start the machine you need:

```
$ virsh --connect qemu:///system start fuss-server
```

- If necessary, start a graphics session on the machine:

```
$ virt-viewer --connect qemu:///system fuss-server
```

13.2.2 Snapshot

To create a snapshot of a machine:

```
virsh -c qemu:///system snapshot-create-as fuss-server <name> "<description>"
```

To see the list of available snapshots:

```
virsh -c qemu:///system snapshot-list fuss-server
```

To return the machine to a snapshot, removing all changes made in the meantime:

```
virsh -c qemu:///system snapshot-revert fuss-server <name>.
```

or, to use the current snapshot (again losing the changes):

```
virsh -c qemu:///system snapshot-revert fuss-server --current
```

13.3 System configurations

13.3.1 Network configuration

Within the fuss-server, network interfaces as defined on this page can be configured by adding the following lines to `/etc/network/interfaces`:

```
allow-hotplug enp1s0
iface enp1s0 inet dhcp

allow-hotplug enp2s0
iface enp2s0 inet static
    address 192.168.6.1
    netmask 255.255.255.0
```

In fuss-client, on the other hand, no configuration is required, since dhcp is used as the default.

13.4 See also.

In case of problems or to learn more about using libvirt from the command line, the page on Libvirt in the Arch Linux wiki is very useful, most of which also applies to Debian or Debian-based systems.

CHAPTER 14

Contribute

Anyone can help improve this documentation that is written in reStructuredText.

CHAPTER 15

Support

If you need help, email info@fuss.bz.it

CHAPTER 16

Licenses