Understanding Debian Packages

A very brief introduction to what's inside Debian binary packages

Miriam Ruiz <miriam@debian.org>

MiniDebConf BCN 2014

Contents

What is a Package?

What is a Package?

- **Minimal Unit of Installation and Removal**
- A Debian package can include:
  - The **files** needed to provide some functionality to the system (software, artwork, configuration files, program data, etc)
  - Some administrative **information** regarding that package: Description, Dependencies, Maintainer, etc.
  - The **copyright/licensing information** about the contents included in that package, though this can't be read until the package is open.
  - **Configuration scripts** to install/deinstall/upgrade/purge the software
  - **Checksums** to check if the files contained in the file have not been modified. Keep in mind that these are not provided for security against attacks.
  - A mechanism (**debconf**) to configure some parameters about the package.

What can you do with a Package?

- **Install** it: `dpkg --install` *`<file.deb>`*
- Show **information** about it: `dpkg --info` *`<file.deb>`*
- List its **contents**: `dpkg --contents` *`<file.deb>`*
- **Extract** its files: `dpkg --extract` *`<file.deb>`* *`<directory>`*
- **Extract and show** files: `dpkg --vextract` *`<file.deb>`* *`<directory>`*

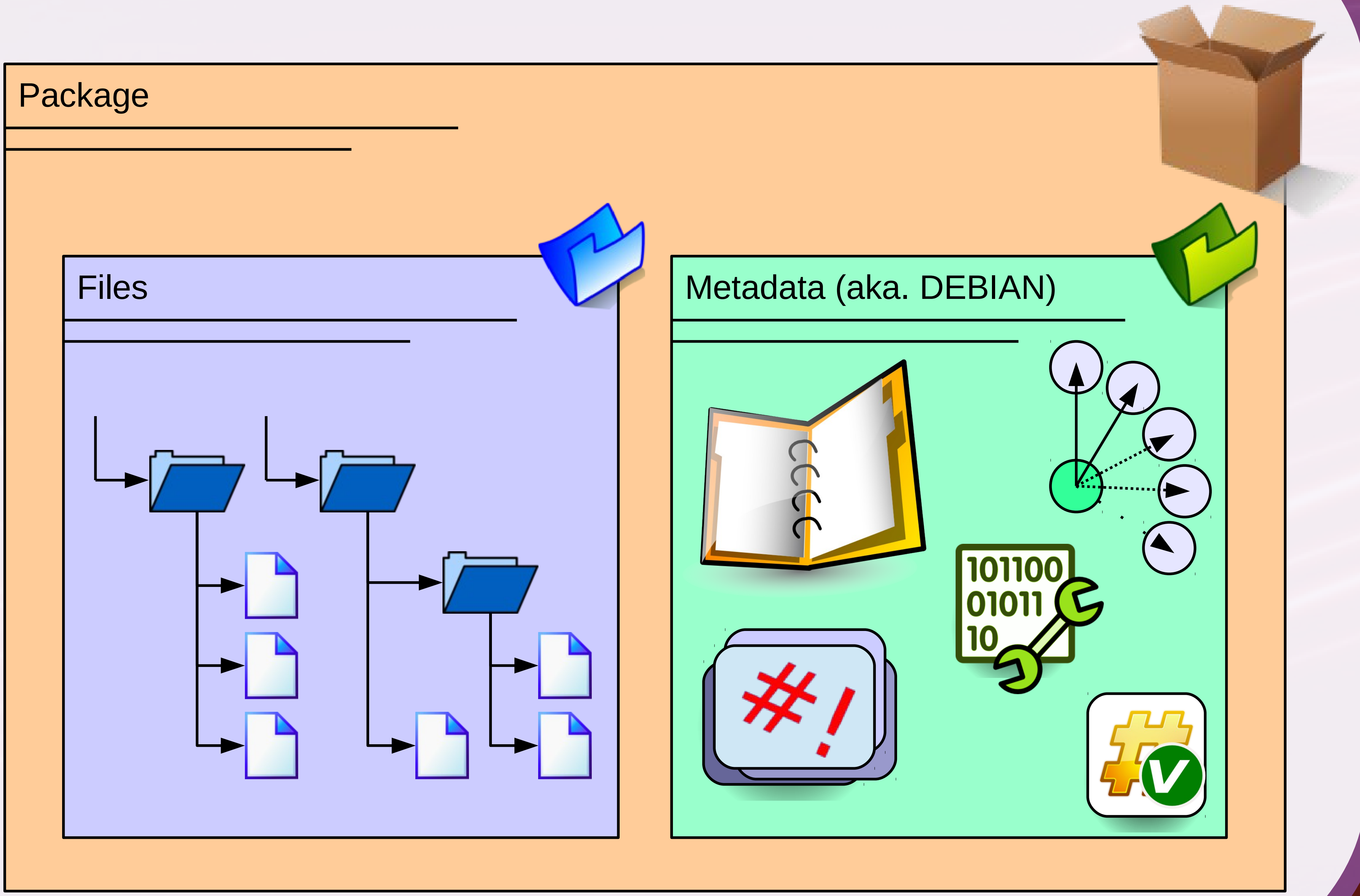
- Lists the **installed packages**: dpkg --list *`<pattern>`*
- List the **files installed** from a package: `dpkg --listfiles` *`<package>`*
- Shows **information** about a package: dpkg `--print-avail` *`<package>`*
- Packages where a given **filename is found**: dpkg `--search` *`<filename>`*
- Shows the **status** of a fiven package: `dpkg --status` *`<package>`*

- **Remove** package: `dpkg --remove` *`<package>`*
- Remove it and **purge** its configuration: `sudo dpkg --purge` *`<package>`*

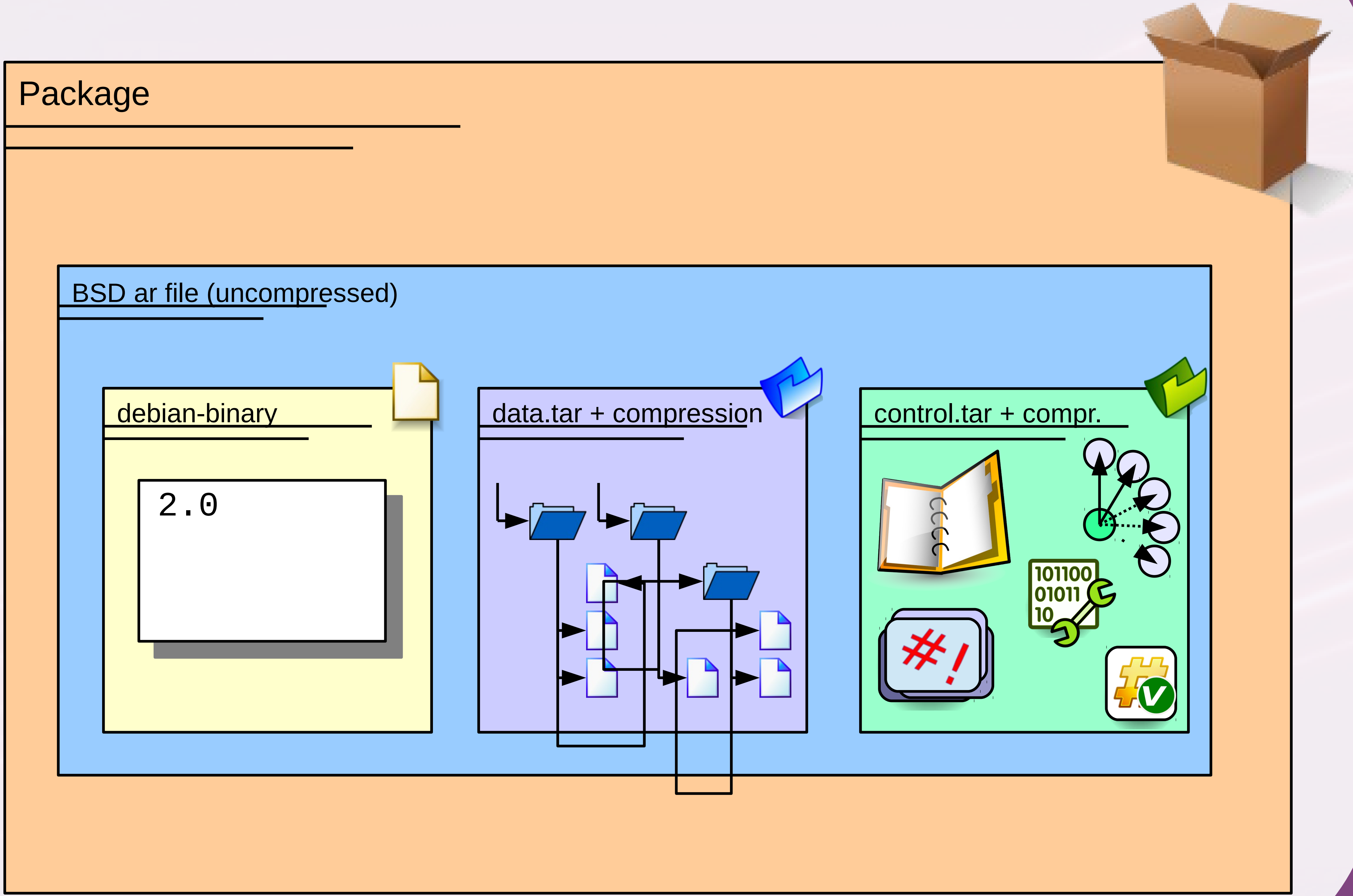
See: https://www.debian.org/doc/manuals/debian-reference/ch02.en.html

What's inside a Package?

How is a package inside?



How are these things stored?



See: https://www.debian.org/doc/manuals/debian-faq/ch-pkg_basics

How can I extract the contents?

- Extract the archived contents: `ar x` *`<file.deb>`*
  - `Result: control.tar.gz data.tar.gz debian-binary`
- Standards version of the binary package : `cat debian-binary`
- Which files are distributed: `tvfJ data.tar.xz`
- Which metadata files are included: `tvfz control.tar.gz`
  - Some files that we can find: `control, prerm. postrm, preinst, postinst, shlibs, conffiles, config, templates, md5sums`
- It is easier to work with the dpkg-deb command:
  - Show **information** about a package: `dpkg --info` *`<file.deb>`*
  - Show a **control field**: `dpkg --field` *`<file.deb>`* *`<field>`*
  - Show a **control file**: `dpkg --info` *`<file.deb>`* *`<control_file>`*
  - Show the **data contents**: `dpkg --contents` *`<file.deb>`*
  - **Extract** all: `dpkg --raw-extract` *`<file.deb>`* *`<directory>`*
  - **Build** a package: `dpkg --build` *`<directory>`* *`<directory>`*

The control data of a package

Control Fields: DEBIAN/control

```
Package: [package name]
Version: [version-deb]
Architecture: [all or architecture_id. Try 'dpkg-architecture -L']
Maintainer: [Name <email@debian.org>]
Installed-Size: [est. inst. size in bytes, div. By 1024, rounded up]
Pre-Depends: [packages needed before even starting the installation]
Depends: [absolute dependency]
Recommends: [strong, but not absolute, dependency]
Suggests: [one package may be more useful with one or more others]
Enhances: [can enhance the functionality of another package]
Breaks: [will not be unpacked unless other package is deconfigured]
Conflicts: [will not be unpacked in the system at the same time]
Provides: [virtual packages]
Replaces: [overwrites files in other packages or replaces them]
Section: [application area]
Priority: [extra, optional, standard, important, required, essential]
Homepage: [web page]
Description: short description
 This is the long description of the package
```

See: https://www.debian.org/doc/debian-policy/ch-controlfields.html
See: https://www.debian.org/doc/manuals/debian-faq/ch-pkg_basics

Relationship with other packages

- **Pre-Depends**: Forces dpkg to complete installation of the packages named before even starting the installation of the package which declares the pre-dependency
- **Depends**: A package will not be configured unless all of the packages listed in its Depends field have been correctly configured (unless there is a circular dependency).
- **Recommends**: A strong, but not absolute, dependency. Packages that would be found together with this one in all but unusual installations.
- **Suggests**: Packages that are related to this one and can perhaps enhance its usefulness, but that installing this one without them is perfectly reasonable.
- **Enhances**: Similar to Suggests but works in the opposite direction. A package can enhance the functionality of another package.
- **Breaks**: The package won't be unpacked unless the broken package is deconfigured first, and it will refuse to allow the broken package to be reconfigured.
- **Conflicts**: The packages won't be unpacked on the system at the same time.
- **Replaces**: If the overwriting package declares that it Replaces the one containing the file being overwritten, then dpkg will replace the file from the old package with that from the new. Normally, Breaks should be used in conjunction with Replaces.
- **Provides**: A virtual package is one which appears in the Provides control field of another package. The effect is as if the package(s) which provide a particular virtual package name had been listed by name everywhere the virtual package name appears.

See: https://www.debian.org/doc/manuals/maint-guide/dother.en.html#conffiles

What are conffiles (DEBIAN/conffiles)?

- List of configuration packages thet dpkg will not overwrite when the package is upgraded
- Those files are usually placed in /etc/
- To determine exactly which files are preserved during an upgrade, for a package you have already installed, you can run:
  - `dpkg --status <package>`
- To see the conffiles in a debian package file, you can do:
  - `dpkg --info <file.deb> conffiles`

See: https://www.debian.org/doc/manuals/maint-guide/dother.en.html#conffiles

Checksums and signatures

- The file `DEBIAN/md5sums` holded a relationship of all the files included in data.tar.xz file and their corresponding MD5 checksum once extracted. This data is **not meant for security reasons**, because they would be easily modifiable, but as a way to check if the files have been modified.
- You can use, for example `debsums -a` *`<package>`*, to check if you have modified a file in your system, that is supposed to be managed by dpkg.
- The security is managed via cryptographic signatures at an APT level, and not at an individual package level.
- You can sign and verify individual packages via debsigs and debsig-verify, or dpkg-sig, although I don't think it is too widely spread. I have never tried it.

See: https://www.debian.org/doc/debian-policy/ap-pkg-controlfields.html
See: https://wiki.debian.org/SecureApt
See: http://purplefloyd.wordpress.com/2009/02/05/signing-deb-packages/

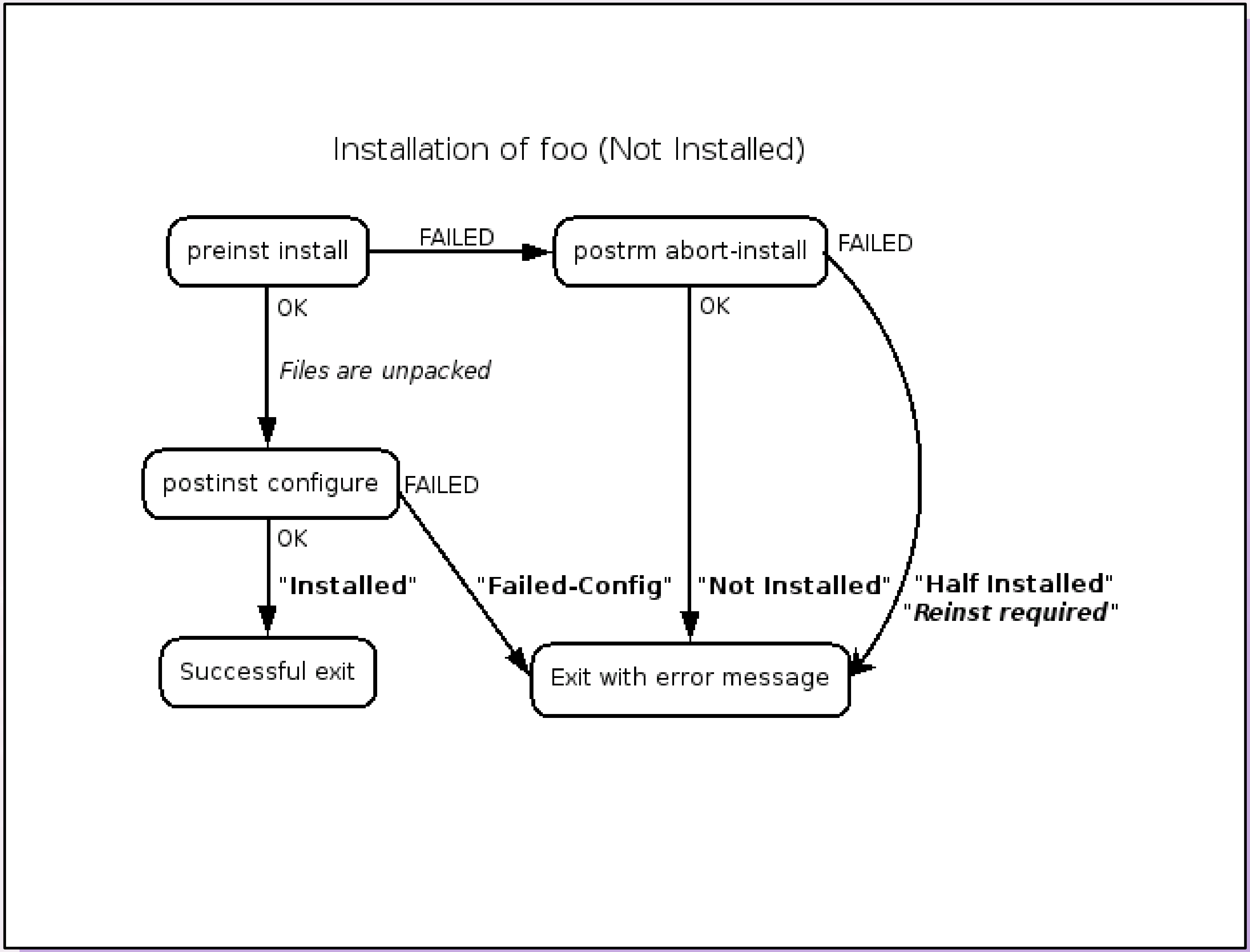
Scripts that handle package changes

preinst, postinst, prerm, and postrm scripts

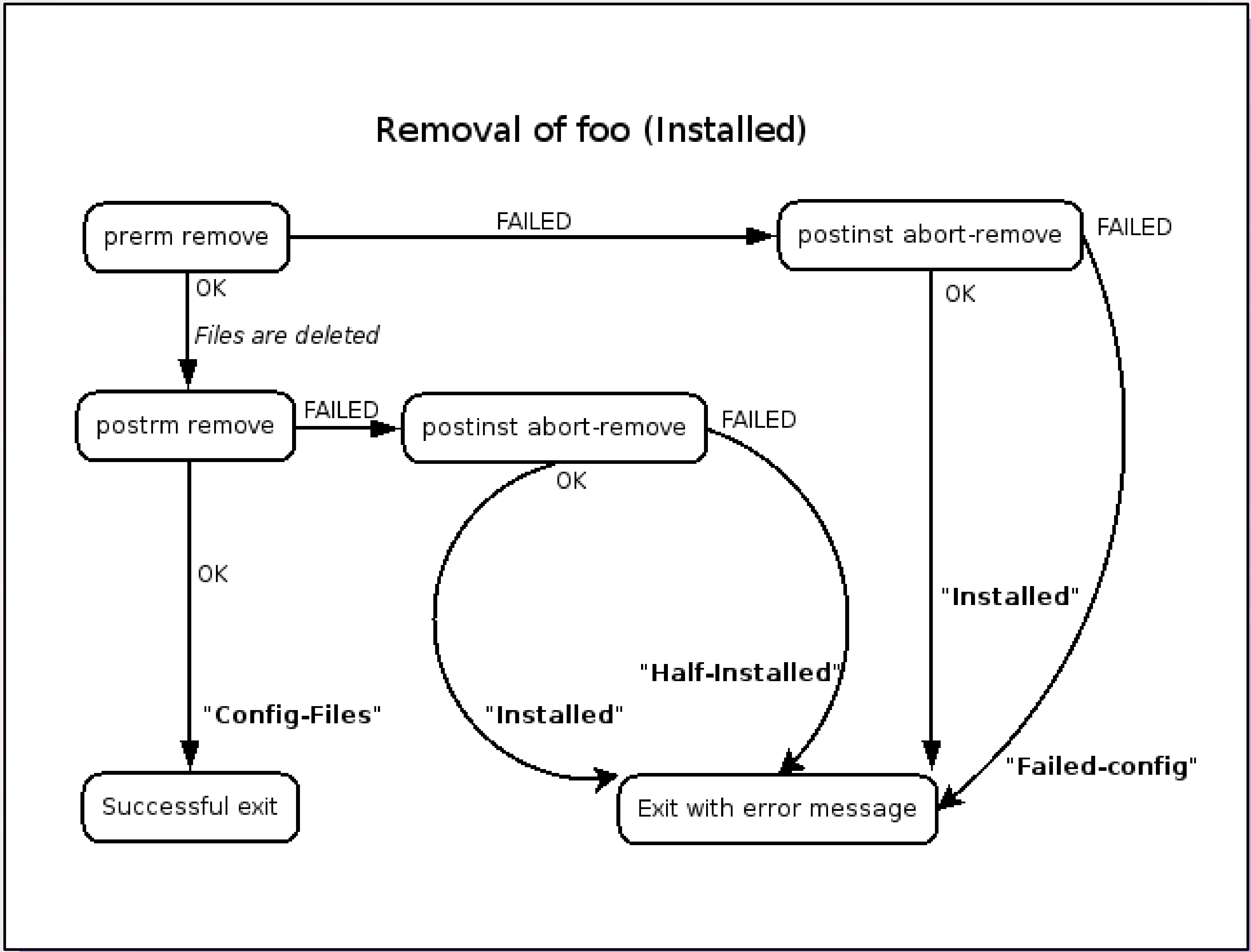
- **Preinst**: This script executes before that package will be unpacked from its Debian archive (".deb") file. Many 'preinst' scripts stop services for packages which are being upgraded until their installation or upgrade is completed.
- **Postinst**: This script typically completes any required configuration of the package foo once foo has been unpacked from its Debian archive (".deb") file. Often, 'postinst' scripts ask the user for input, and/or warn the user that if they accept default values, they should remember to go back and reconfigure that package afterwards. Many 'postinst' scripts execute any commands necessary to start or restart a service once a new package has been installed or upgraded.
- **Prerm**: This script typically stops any daemons which are associated with a package. It is executed before the removal of files associated with the package.
- **Postrm**: This script typically modifies links or other files associated with foo, and/or removes files created by the package.

See: https://www.debian.org/doc/manuals/debian-faq/ch-pkg_basics
See: http://www.marga.com.ar/~marga/debian/diagrams/
See: http://people.debian.org/~srivasta/MaintainerScripts.html

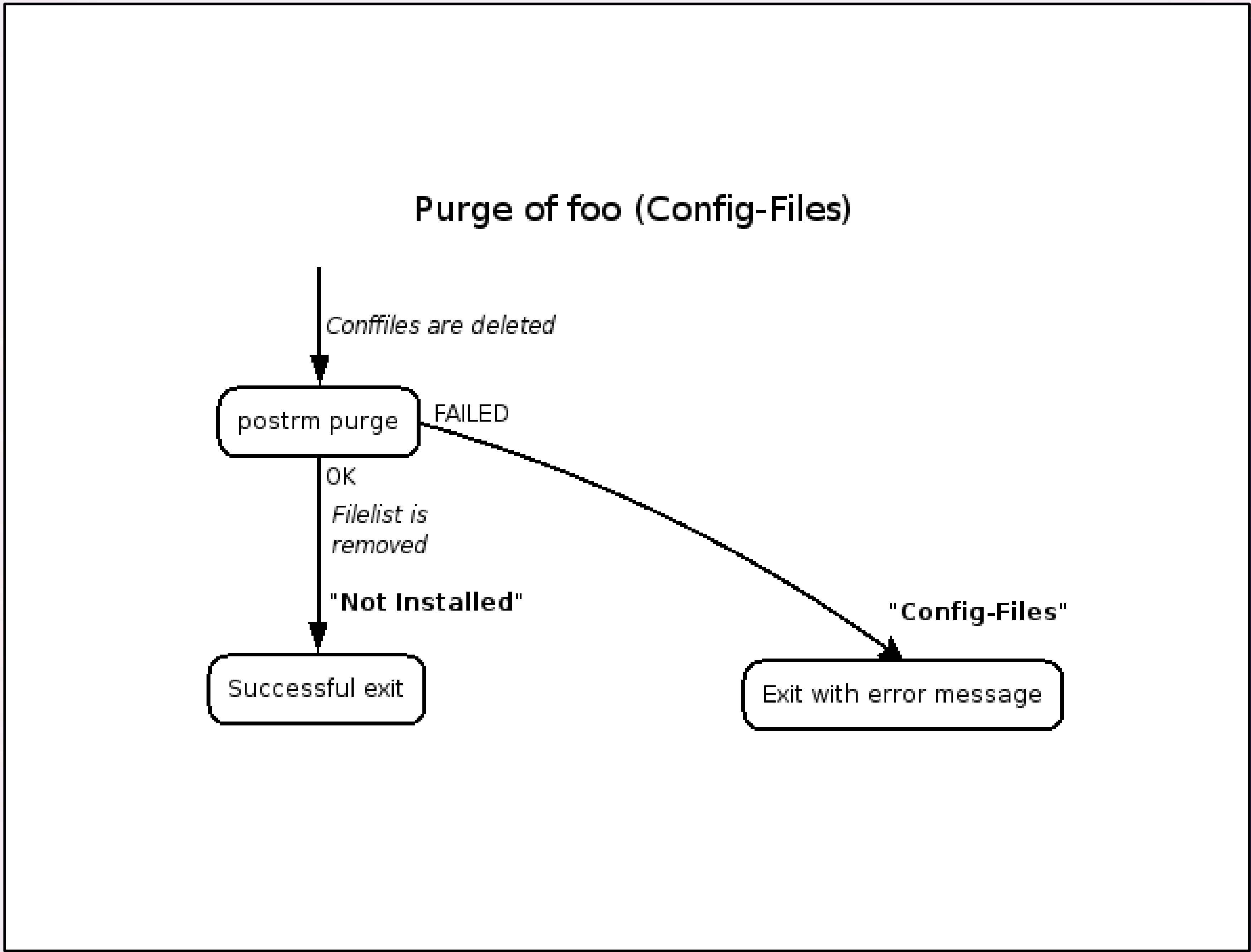
Installing a package



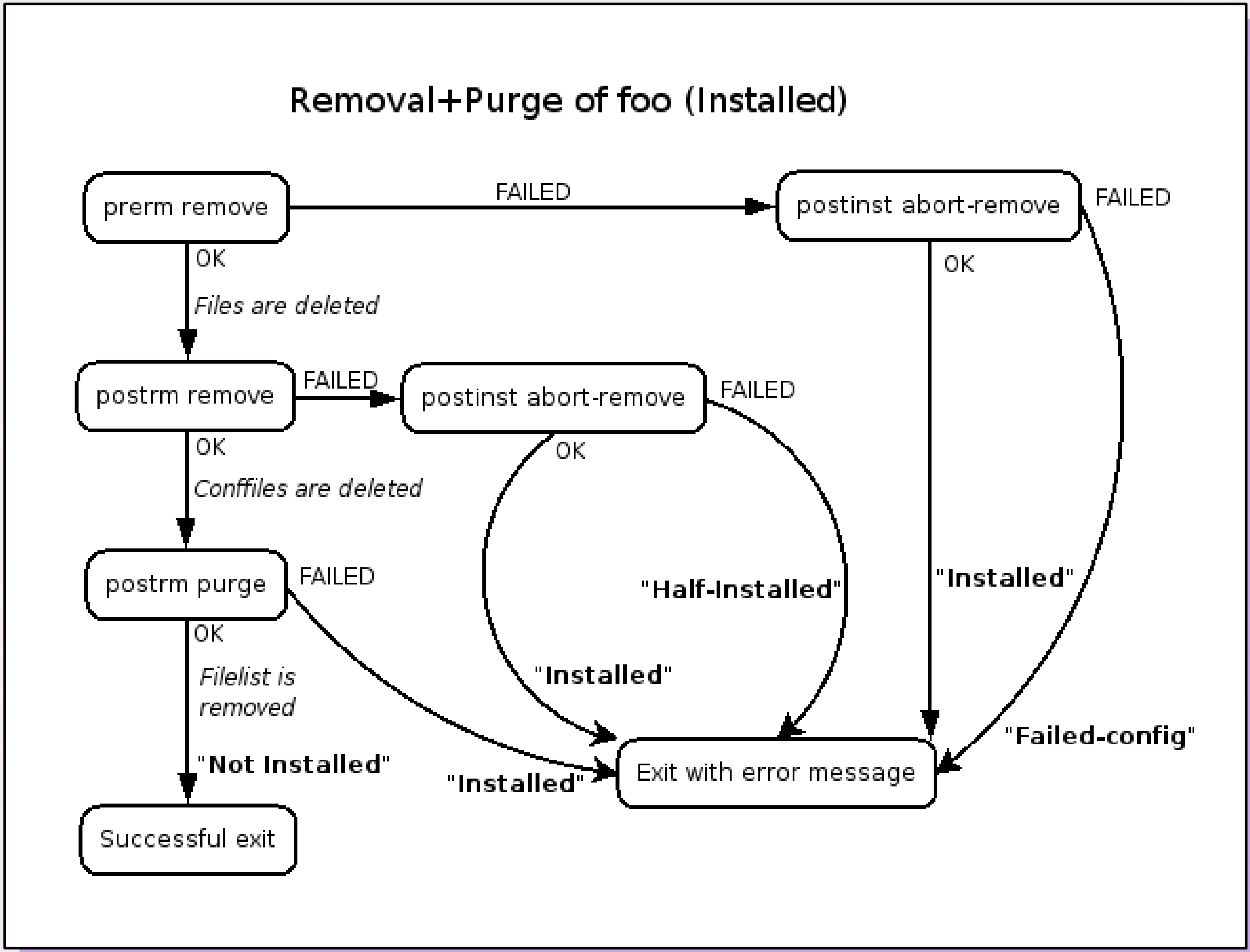
Removing a package



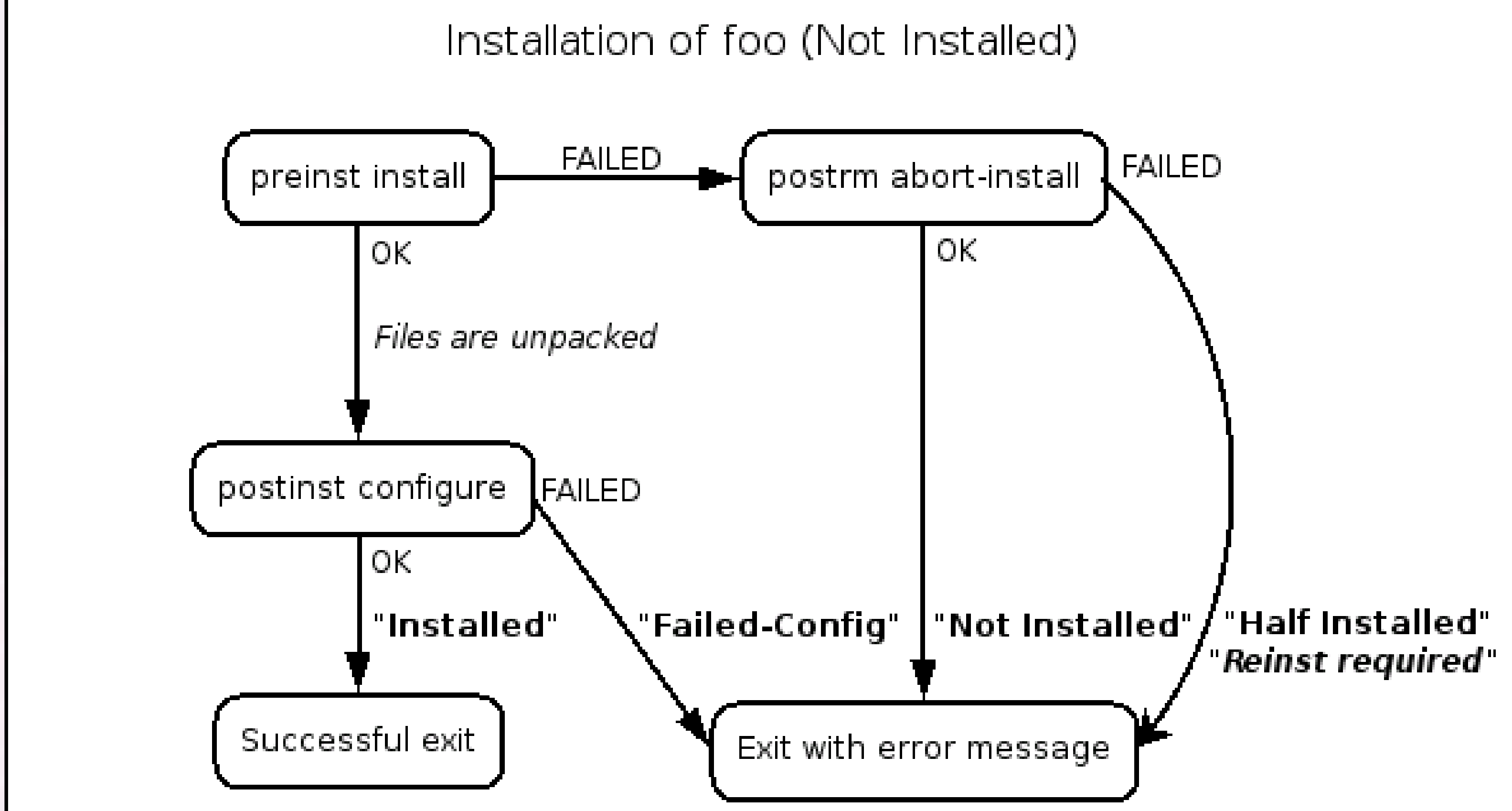
Purging a package



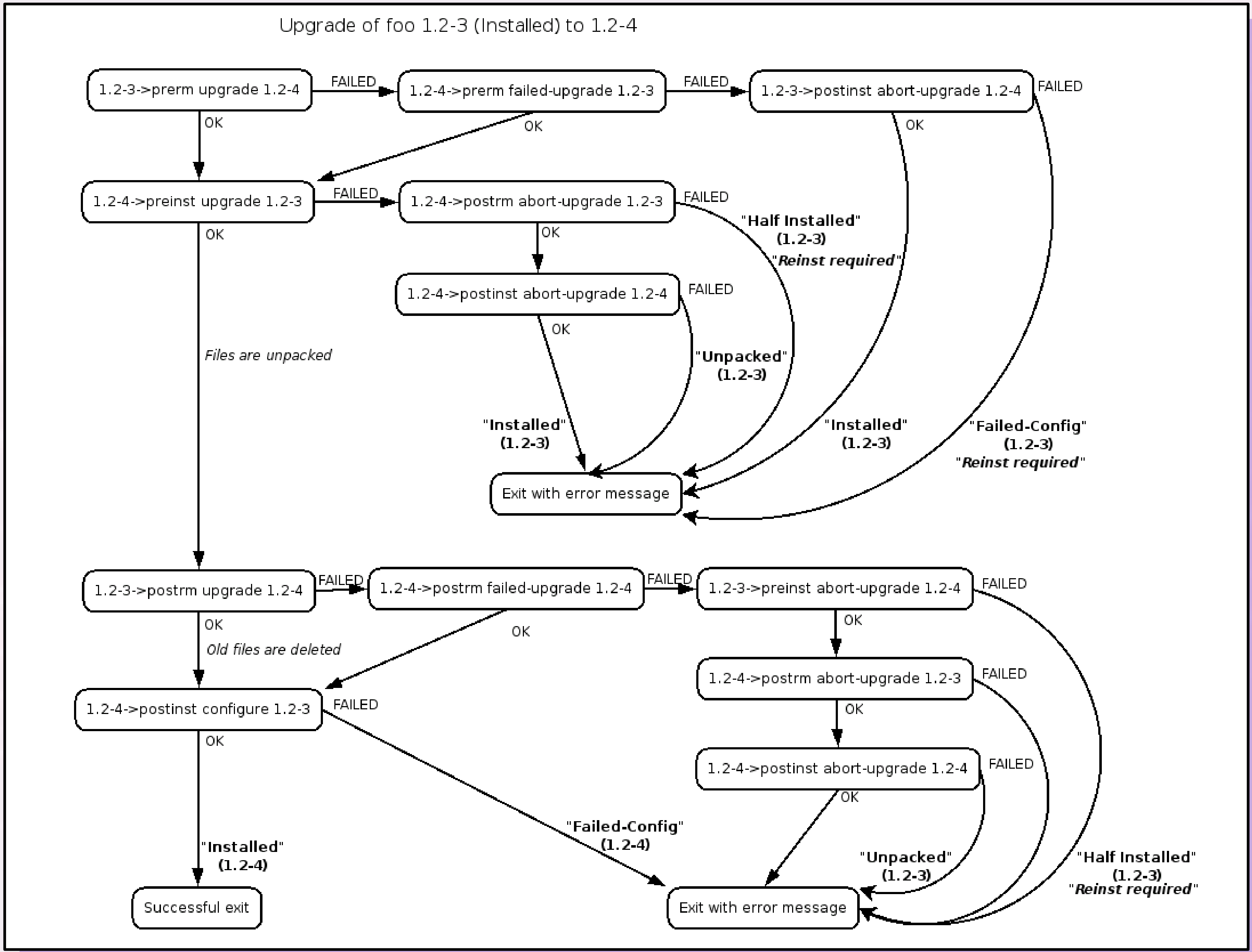
Removing and purging a package



Installing an already configured package



Upgrading a package


Upgrade of foo 1.2-3 (Installed) to 1.2-4
1.2-3->prerm upgrade 1.2-4
FAILED
1.2-4->prerm failed-upgrade 1.2-3
FAILED
1.2-3->postinst abort-upgrade 1.2-4
FAILED
OK
OK
OK
1.2-4->preinst upgrade 1.2-3
FAILED
1.2-4->postrm abort-upgrade 1.2-3
FAILED
"Half Installed" (1.2-3) "Reinst required"
OK
OK
1.2-4->postinst abort-upgrade 1.2-4
FAILED
OK
Files are unpacked
"Unpacked" (1.2-3)
"Installed" (1.2-3)
"Installed" (1.2-3)
"Failed-Config" (1.2-3) "Reinst required"
Exit with error message
1.2-3->postrm upgrade 1.2-4
FAILED
1.2-4->postrm failed-upgrade 1.2-4
FAILED
1.2-3->preinst abort-upgrade 1.2-4
FAILED
OK
OK
Old files are deleted
OK
1.2-4->postinst configure 1.2-3
FAILED
1.2-4->postrm abort-upgrade 1.2-3
FAILED
OK
1.2-4->postinst abort-upgrade 1.2-4
FAILED
OK
OK
"Failed-Config" (1.2-4)
"Installed" (1.2-4)
"Unpacked" (1.2-3)
"Half Installed" (1.2-3) "Reinst required"
Successful exit
Exit with error message

DPKG Triggers

- Triggers are used to ensure that during the standard package-management process **certain operations always run, but not more than necessary**.
- Trigger-using packages can be classified in two behavioral categories:
  - Consumers: packages which declare triggers and thus can be "triggered"
  - Producers: packages which activate triggers (explicitly or implicitly)
- When a consumer is triggered, its postinst script is run with the arguments:
  - `postinst triggered "<trigger1> ... <triggerN>"`

See: http://www.seanius.net/blog/2009/09/dpkg-triggers-howto/
See: http://stackoverflow.com/questions/15276535/dpkg-how-to-use-trigger

States in which Debian Packages can be

- **Not-installed**: The package is not installed on your system.
- **Config-files**: Only the configuration files of the package exist on the system.
- **Half-installed**: The installation of the package has been started, but not completed for some reason.
- **Unpacked**: The package is unpacked, but not configured.
- **Half-configured**: The package is unpacked and configuration has been started, but not yet completed for some reason.
- **Triggers-awaited**: The package awaits trigger processing by another package.
- **Triggers-pending**: The package has been triggered.
- **Installed**: The package is unpacked and configured OK.

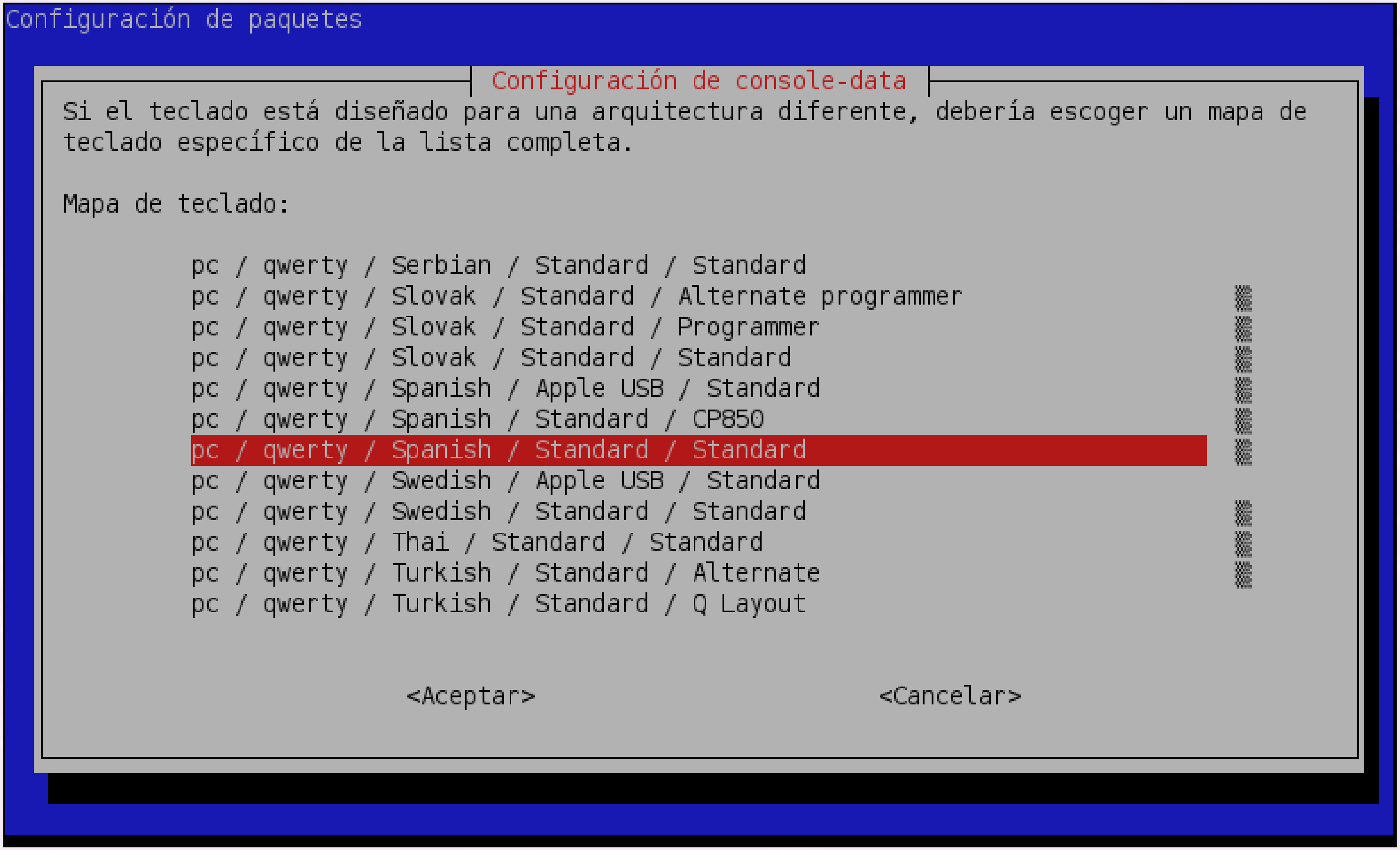
Debian Configuration: Debconf

Debian Configuration: Debconf

- Debconf is a **backend database**, with a frontend that talks to it and presents an interface to the user. There can be many different types of frontends, from plain text to a web frontend.
- The frontend also talks to a special config script in the control section of a debian package, and it **can talk to postinst scripts and other scripts as well**, all using a special protocol. These scripts tell the frontend what values they need from the database, and the frontend asks the user questions to get those values if they aren't set.
- All the configuration information is stored in a **special database** that defines a **hierarchy of information**. Each package receives its own space in the hierarchy and is free to use a flat space, or divide its space further into sub-hierarchies.
- If multiple packages share a common purpose they may use a shared toplevel hierarchy, preferably with the same name as a **shared (virtual) package name**.
- Each **variable** in the configuration space has some **information** associated with it, and it has a **value**.

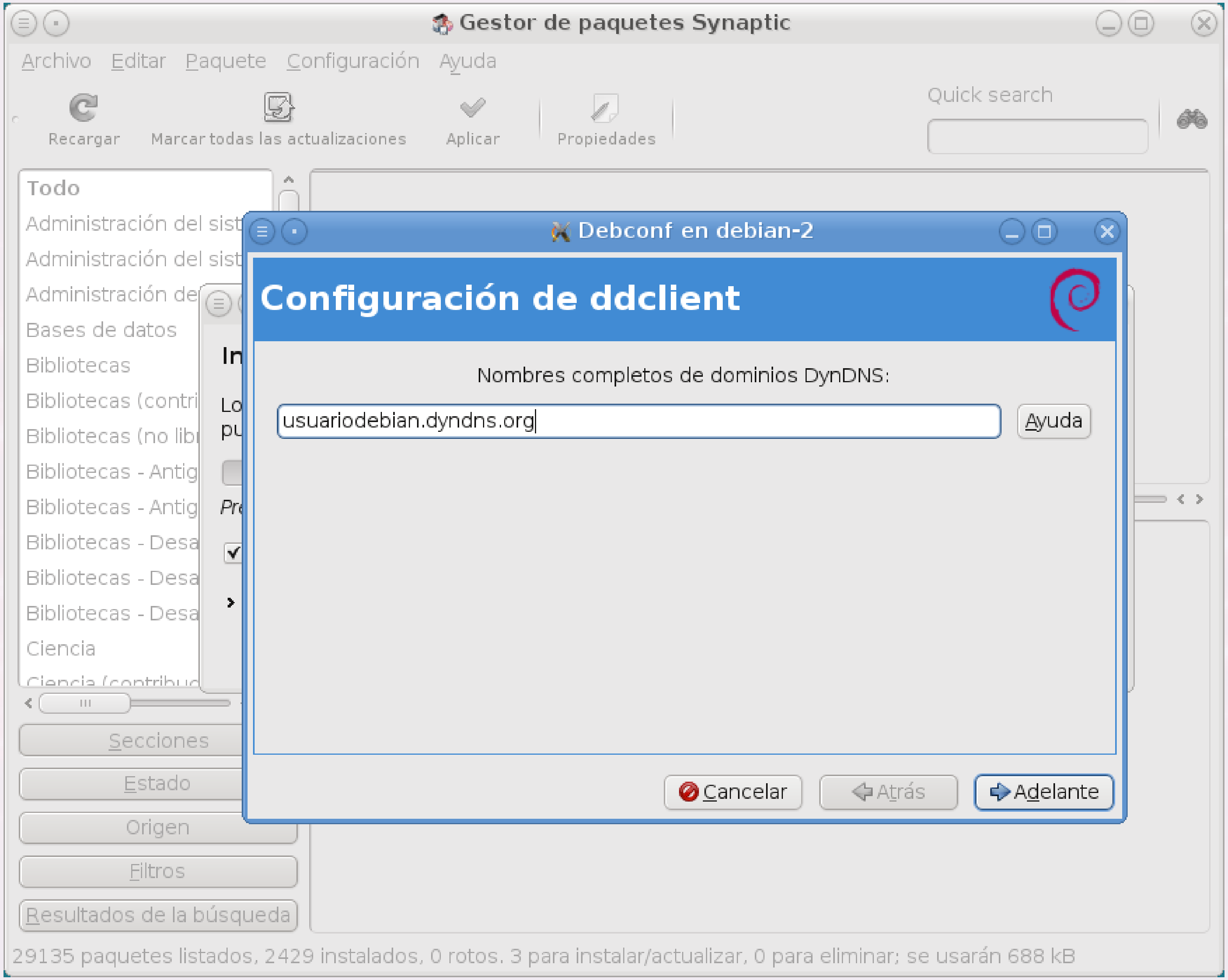
See: http://www.fifi.org/doc/debconf-doc/tutorial.html
See: https://www.debian.org/doc/packaging-manuals/debconf_specification.html

Debian Configuration: Debconf



```
Configuración de paquetes

                        Configuración de console-data
 Si el teclado está diseñado para una arquitectura diferente, debería escoger un mapa de
 teclado específico de la lista completa.

 Mapa de teclado:

          pc / qwerty / Serbian / Standard / Standard
          pc / qwerty / Slovak / Standard / Alternate programmer
          pc / qwerty / Slovak / Standard / Programmer
          pc / qwerty / Slovak / Standard / Standard
          pc / qwerty / Spanish / Apple USB / Standard
          pc / qwerty / Spanish / Standard / CP850
          pc / qwerty / Spanish / Standard / Standard
          pc / qwerty / Swedish / Apple USB / Standard
          pc / qwerty / Swedish / Standard / Standard
          pc / qwerty / Thai / Standard / Standard
          pc / qwerty / Turkish / Standard / Alternate
          pc / qwerty / Turkish / Standard / Q Layout


               <Aceptar>                                <Cancelar>
```

Debian Configuration: Debconf



Configuración de ddclient

Nombres completos de dominios DynDNS:

usuariodebian.dyndns.org

29135 paquetes listados, 2429 instalados, 0 rotos. 3 para instalar/actualizar, 0 para eliminar; se usarán 688 kB

Debconf script: DEBIAN/config

```
#!/bin/sh

set -e

. /usr/share/debconf/confmodule

if [ -f /usr/share/dbconfig-common/dpkg/config.mysql ]; then
        . /usr/share/dbconfig-common/dpkg/config.mysql
        if ! dbc_go phpmyadmin $@ ; then
            echo 'Automatic configuration using dbconfig-common
failed!'
        fi
fi

db_version 2.0

db_input high phpmyadmin/reconfigure-webserver || true

if [ ! -f /etc/phpmyadmin/htpasswd.setup ]; then
    db_input low phpmyadmin/setup-username || true
    db_input low phpmyadmin/setup-password || true
fi

db_go || true
```

See: https://www.debian.org/doc/packaging-manuals/debconf_specification.html

Debconf templates: DEBIAN/templates

```
Template: hostname
Type: string
Default: debian
Description: unqualified hostname for this computer
 This is the name by which this computer will be known on the network.
It
 has to be a unique name in your domain.

Template: domain
Type: string
Description: domain for this computer
  This is the domain your computer is a member of. Typically it is
  something like "mycompany.com" or "myuniversity.edu".
```

See: https://www.debian.org/doc/packaging-manuals/debconf_specification.html

Configuration templates: DEBIAN/templates

- Debconf is a **backend database**, with a frontend that talks to it and presents an interface to the user. There can be many different types of frontends, from plain text to a web frontend.
- The frontend also talks to a special config script in the control section of a debian package, and it **can talk to postinst scripts and other scripts as well**, all using a special protocol. These scripts tell the frontend what values they need from the database, and the frontend asks the user questions to get those values if they aren't set.
- All the configuration information is stored in a **special database** that defines a **hierarchy of information**. Each package receives its own space in the hierarchy and is free to use a flat space, or divide its space further into sub-hierarchies.
- If multiple packages share a common purpose they may use a shared toplevel hierarchy, preferably with the same name as a **shared (virtual) package name**.
- Each **variable** in the configuration space has some **information** associated with it, and it has a **value**.
- You can see its contents using the command: `debconf-show`

See: http://www.fifi.org/doc/debconf-doc/tutorial.html
See: https://www.debian.org/doc/packaging-manuals/debconf_specification.html

Debian Configuration: Debconf

- Debconf is a backend database, with a frontend that talks to it and presents an interface to the user. There can be many different types of frontends, from plain text to a web frontend.
- The frontend also talks to a special config script in the control section of a debian package, and it can talk to postinst scripts and other scripts as well, all using a special protocol. These scripts tell the frontend what values they need from the database, and the frontend asks the user questions to get those values if they aren't set.

See: http://www.fifi.org/doc/debconf-doc/tutorial.html
See: https://www.debian.org/doc/packaging-manuals/debconf_specification.html

Question time!

Some Links

- Debian Policy Manual:
  - https://www.debian.org/doc/debian-policy/
- Basics of the Debian Package Management System:
  - https://www.debian.org/doc/manuals/debian-faq/ch-pkg_basics
- Debian Administrator's Handbook:
  - http://debian-handbook.info/browse/stable/sect.manipulating-packages-with-dpkg.html
- Getting Information About Packages with APT:
  - http://newbiedoc.sourceforge.net/tutorials/apt-get-intro/info.html
- Maintainer Scripts:
  - http://people.debian.org/~srivasta/MaintainerScripts.html
- Debconf Specification:
  - https://www.debian.org/doc/packaging-manuals/debconf_specification.html
- The Debconf Programmer's Tutorial
  - http://www.fifi.org/doc/debconf-doc/tutorial.html
-

Understanding Debian Packages

A very brief introduction to what's inside Debian binary packages

Miriam Ruiz <miriam@debian.org>

MiniDebConf BCN 2014