How to Create DEB Packages for Debian and Ubuntu

The DEB file in Debian-based operating systems like Ubuntu and Kali Linux is equivalent to the EXE file found in Windows.

The DEB package is an archive containing all the files including the compiled version of the application, source code, configuration files, images, and installation commands. The DEB file in Debian-based operating systems like Ubuntu and Kali Linux is equivalent to the EXE file found in Windows.

Here's how you can develop your own DEB packages for a Debian-based Linux distribution.

Step 1: Install required packages

Preparing a Debian package requires some programming. To get started, install these utilities on your system:

```
sudo apt install build-essential binutils lintian debhelper dh-make devscripts
```

Step 2: Select package

Before creating a Debian package (DEB) for a program, you should consider several points:

Check if the package you intend to create is already in the Debian repositories:

```
apt-cache search package-name
```

Check the license type that the program you will be packing. There is a general rule about the use of the GNU/GPL license.

Make sure that the program does not cause security problems for the system.

Contact the author of the program. Notify Debian developers to include this program in the Debian repository.

Step 3: Start preparing the package

First, create a new folder in your home directory to avoid confusion.

```
cd /home mkdir package cd package
```

Then unzip the tar archive containing the source code of the program you will be creating the package in this directory. To illustrate, the example will use the rsyslog archive.

```
tar -zxvf rsyslog-6.3.6.tar.gz
```

Navigate to the newly created directory with the cd command:

```
cd rsyslog-6.3.6
```

Usually, program source code comes with **INSTALL** and **README files.** Even if you know what a program is and how it works, it can be very beneficial to spend some time reading these files.

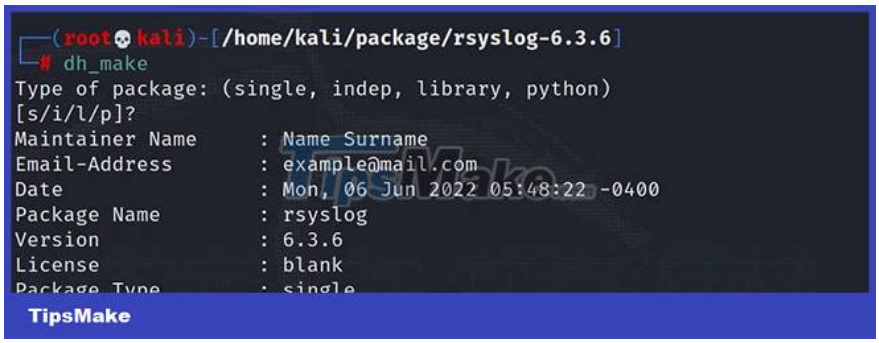
There are commands like **./configure make** and **make install** that can easily install those repositories on the system. But there are some parameters to the **./configure** option that you should know about. You can use the command **./configure --help** to get such information.

Step 4: Add developer information

Before creating a DEB package for your program, pay attention to the package name and version number. You will also need some additional information about the packager when creating the package. For this you have to export your information using the following commands:

```
export DEBEMAIL="[email protected]" export DEBFULLNAME="Name Lastname" After this
```

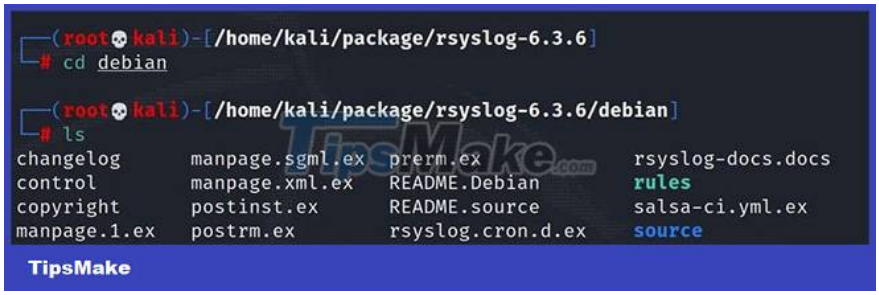
After issuing the dh_make command, you should select your package type and press **Enter.**



Following this step, you will see a folder in the parent folder with the extension **".orig".** If this doesn't work, try running the **dh_make command with the --createorig** parameter .

```
ls # Output rsyslog-6.3.6 rsyslog_6.3.6.orig.tar.xz rsyslog-6.3.6.tar.gz
```

You may also see a new directory named Debian in the current working directory. These folders and files contain all information regarding the Debian package about the program.

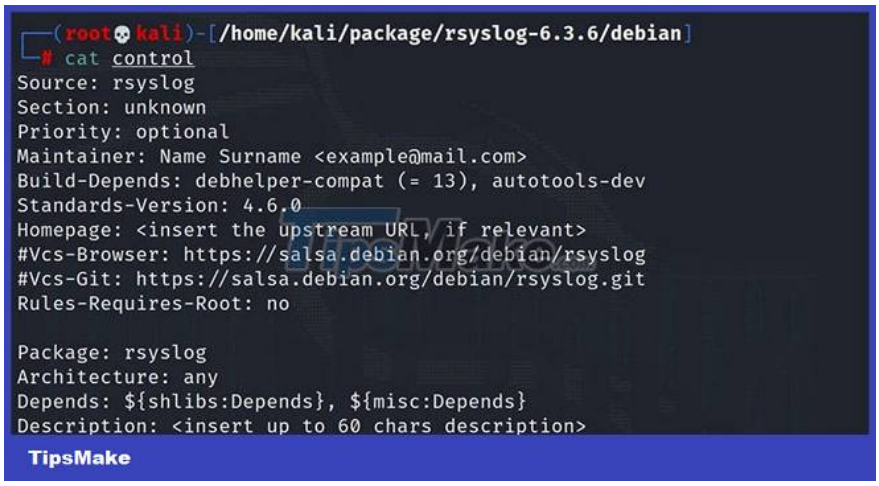


Step 5: Debian files related to the package

You need to know the following information about the files located in the Debian directory.

1. File control

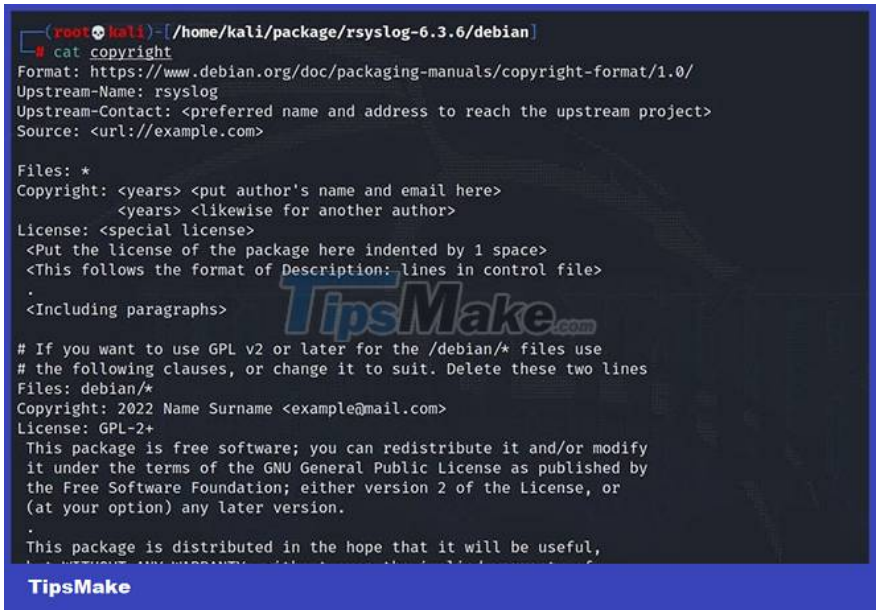
The file control provides a lot of information related to the package.



1. **Source:** The line where you will specify your program name
2. **Section:** The line that identifies the part your program belongs under the license
3. **Maintainer:** The line containing the information of the person who prepared the package
4. **Build-Depends:** The dependencies are listed on this line
5. **Depends:** This line is very important. You specify the dependencies of the package with this value
6. **Description:** Line where you can enter information about the package

2. File copyright

This file contains information about the program's license. Its default content is as follows:



3. File changelog

This file is like your program log route. If you've done something independent of the program's source or if you've fixed some bugs, you can add it to this file.

4. File rules

The rules file is like a Makefile for your Debian package. When installing a prepared Debian package with dpkg, the information in this file is taken as a basis.

Of course, you can change the parameters in this file as you see fit.

5. Other files in the folder

It may also be helpful to know the functions of the following files:

1. **README.Debian:** File Readme
2. **conffiles.ex:** Use this file if you want to keep your old installation files while installing the program
3. **cron.d.ex:** You can perform cron operations using this file
4. **dirs:** Use this file to specify directories that are not installed during installation but will be created later
5. **docs:** If there are documents included with your program, specify them with this file
6. **emacsen*.ex:** If your program needs the Emacs file during installation, specify it with this file
7. **init.d.ex:** Use this file if you want your program to run at system startup

To continue to the next stage, delete any files you believe you no longer need. Then, rename the file extensions and remove **the ".ex"** at the end. ".Ex" (example) indicates that this is a sample file.

Step 6: Build the package

If you've done this, you can now prepare a Debian package for your program. For this, run the following command:

```
dpkg-buildpackage
```

Another important issue here is creating a GPG for the e-mail address that you export as **Maintainer.**

```
export DEBEMAIL="[email protected]"
```

dpkg will look up your GPG info while creating the package. You can list it with the command **gpg --list-key** .

If you have any problems during the dpkg-buildpackage phase, try the following command:

```
dpkg-buildpackage -nc -i
```

This command will skip some parts that can cause errors.

If all goes well, the Debian package for your program will be ready to be installed and stored in the next directory. With below command you can install, test and review the package.

```
dpkg -i package-name
```

The DEB package generation system is one of the most fundamental factors that distinguishes Debian as the leader of GNU/Linux. Debian is a big system and it's really important for contributors to be able to create their own packages.

If you are new to GNU/Linux, this may seem confusing. However, as you can see, preparing a Debian package is simpler than you think. Of course, building a Debian package takes time and work.

But that doesn't mean you have to manually create packages for the programs you want to install. There are several sites on the Internet from where you can download free DEB packages.

You finished reading the article "**How to Create DEB Packages for Debian and Ubuntu**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.
