www.debian.org (https://www.debian.org/)

Variant of my Linux Debian 12.1 "Bookworm" Xfce settings +

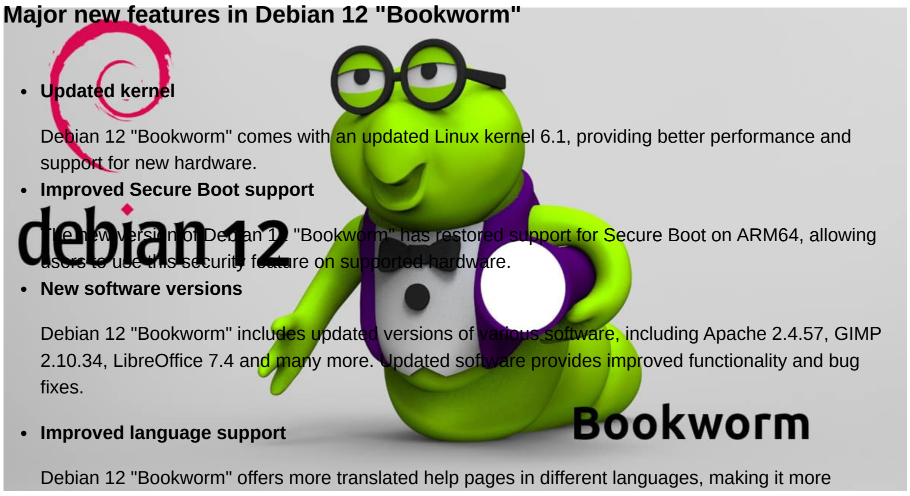
Введение

In this article, I would like to share my personal experience using the Debian 12.1 "Bookworm" operating system with the Xfce workspace. I'll talk about my impressions and the reasons why I chose this configuration. Debian 12 "Bookworm" is the new stable version of Debian that has been released after a long development period. Xfce is a lightweight and fast workspace that is one of the default options in Debian. Let's look at the details of this configuration and the benefits it offers.

Debian 12 "Bookworm"

Debian 12 "Bookworm" is the new stable version **of Debian** that has been released after a long development period. It offers many new features, improvements and updates, making Debian even more powerful and easy to use.

Major new features in Debian 12 "Bookworm"



- **Updated kernel**

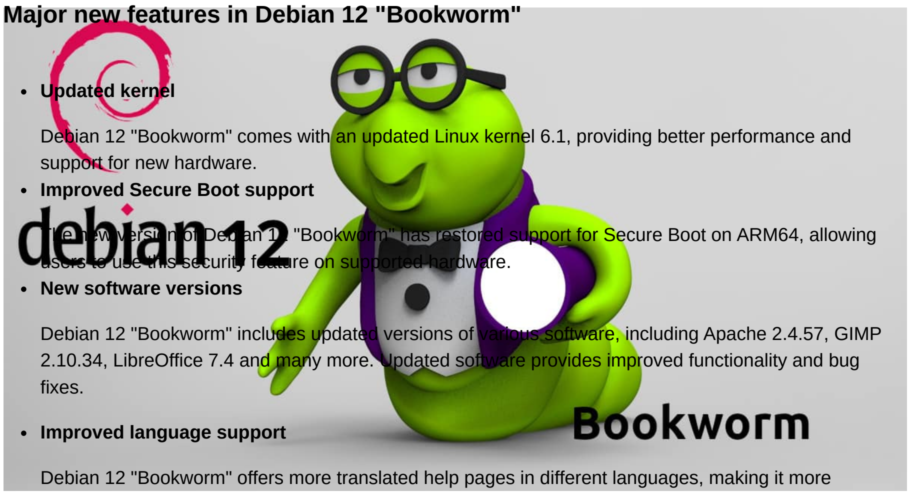
  Debian 12 "Bookworm" comes with an updated Linux kernel 6.1, providing better performance and support for new hardware.
- **Improved Secure Boot support**

  The new version of Debian 12 "Bookworm" has restored support for Secure Boot on ARM64, allowing users to use this security feature on supported hardware.
- **New software versions**

  Debian 12 "Bookworm" includes updated versions of various software, including Apache 2.4.57, GIMP 2.10.34, LibreOffice 7.4 and many more. Updated software provides improved functionality and bug fixes.
- **Improved language support**

  Debian 12 "Bookworm" offers more translated help pages in different languages, making it more accessible to users around the world.

Major new features in Debian 12 "Bookworm"



- **Updated kernel**

  Debian 12 "Bookworm" comes with an updated Linux kernel 6.1, providing better performance and support for new hardware.
- **Improved Secure Boot support**

  The new version of Debian 12 "Bookworm" has restored support for Secure Boot on ARM64, allowing users to use this security feature on supported hardware.
- **New software versions**

  Debian 12 "Bookworm" includes updated versions of various software, including Apache 2.4.57, GIMP 2.10.34, LibreOffice 7.4 and many more. Updated software provides improved functionality and bug fixes.
- **Improved language support**

  Debian 12 "Bookworm" offers more translated help pages in different languages, making it more accessible to users around the world.

Xfce 4.18 workspace

[www.xfce.org (https://www.xfce.org/)](https://www.xfce.org/)

Xfce is a fast and lightweight workspace that provides the user with a convenient and flexible environment for working with the operating system. It is one of the most popular workspaces in the Linux community and is widely used on various distributions, including Debian. Xfce offers a simple and intuitive interface that allows users to customize and tailor their work environment to suit their needs.

Xfce benefits:

- *Lightness.*

  Xfce consumes less system resources compared to other workspaces, making it an ideal choice for older and slower computers. It provides a fast and responsive experience even on devices with limited resources.
- *Flexibility* .

  Xfce offers a wide range of settings and options to personalize your workspace. Users can customize taskbars, menus, hotkeys, and themes to their liking.
- *Stability* .

  Xfce is known for its reliability and stability. It offers smooth and uninterrupted performance, which is especially important for professionals who require a stable environment in which to work.

New features in Xfce 4.18

Xfce **4.18** introduces several new features and improvements to the workspace.

- Release 4.18 includes a new file entry widget and shortcut editor.
- Tumbler has performance improvements and now supports shared **thumbnail** repositories.
- The panel includes a new clock plugin with different layout options.
- **The xfdesktop** manager allows users to hide the "Delete" option from the context menu and adds a confirmation prompt for rearranging desktop icons.
- **The xfce4-settings** settings manager has a simplified search bar and improvements to display and appearance settings.
- **Thunar** is a file manager that has new features such as displaying the number of files in a directory and the date the files were created, as well as two new preview modes for images.
- Also added is the ability to undo and redo operations on files and highlight files in different colors.
- The toolbar can now be customized as desired.

My Debian 12 "Bookworm" configuration with Xfce

Debian 12 "Bookworm" c Xfce

I chose **Debian 12 "Bookworm"** with Xfce workspace for my configuration because this combination offers me the optimal combination of performance and functionality. Xfce gives me ease of use and flexibility to customize my workspace, and Debian 12 "Bookworm" gives me stability and a wide variety of software

Xfce offers a wide range of settings and options to personalize your workspace. Users can customize taskbars, menus, hotkeys, and themes to their liking.

- *Stability* :

Xfce is known for its reliability and stability. It offers smooth and uninterrupted performance, which is especially important for professionals who require a stable environment in which to work.

New features in Xfce 4.18

Xfce **4.18** introduces several new features and improvements to the workspace.

- Release 4.18 includes a new file entry widget and shortcut editor.
- Tumbler has performance improvements and now supports shared **thumbnail** repositories.
- The panel includes a new clock plugin with different layout options.
- **The xfdesktop** manager allows users to hide the "Delete" option from the context menu and adds a confirmation prompt for rearranging desktop icons.
- **The xfce4-settings** settings manager has a simplified search bar and improvements to display and appearance settings.
- **Thunar** is a file manager that has new features such as displaying the number of files in a directory and the date the files were created, as well as two new preview modes for images.
- Also added is the ability to undo and redo operations on files and highlight files in different colors.
- The toolbar can now be customized as desired.

My Debian 12 "Bookworm" configuration with Xfce

Debian 12 "Bookworm" c Xfce

I chose **Debian 12 "Bookworm"** with Xfce workspace for my configuration because this combination offers me the optimal combination of performance and functionality. Xfce gives me ease of use and flexibility to customize my workspace, and Debian 12 "Bookworm" gives me stability and a wide variety of software.

Installation process

Installing Debian 12 "Bookworm" with the Xfce workspace was simple and intuitive. I downloaded the Debian 12 "Bookworm" installation image from the official Debian website and burned it onto a USB drive. I then rebooted my computer from the USB drive and followed the installer's instructions.

During installation, I selected the Xfce workspace as my preferred option. The installer asked me to select system components and settings, and I selected those that suit my needs. After the installation was complete, I rebooted my computer and logged into my new system.

Example settings after a minimal network installation of Debian 12 with an active root account (without sudo):

1. Replace repositories in sources.list

Editing with commands:

```
~$ su
```

Password: enter the root password

In the directory that opens, edit with root rights `sources.list` in nano, vim or any text editor:

Official repositories:

Standard contents of the **sources.list** configuration file :

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

- Let's comment out the following line and save the file with the changes:

```
deb http://deb.debian.org/debian/ bookworm main contrib non-free non-free-firmw>
deb-src http://deb.debian.org/debian/ bookworm main contrib non-free non-free-f>

# deb cdrom:[Debian GNU/Linux 12.0.0 _Bookworm_ - Official amd64 DVD Binary-1 wi>
deb http://security.debian.org/debian-security bookworm-security main contrib n>
deb-src http://security.debian.org/debian-security bookworm-security main contr>
```

- If you wish, you can remove everything from the file by inserting only the following repositories:

```
deb http://deb.debian.org/debian/ bookworm-updates main contrib non-free non-fr>
deb-src http://deb.debian.org/debian/ bookworm-updates main contrib non-free no>
```

We should replace the list of repositories in our system only when it is really necessary, in cases where

In the directory that opens, edit with root rights `sources.list` in nano, vim or any text editor:

Official repositories:

Standard contents of the **sources.list** configuration file :

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

```
deb http://deb.debian.org/debian/ bookworm main contrib non-free non-free
-firmw>
deb-src http://deb.debian.org/debian/ bookworm main contrib non-free non-
free-f>

# deb cdrom:[Debian GNU/Linux 12.0.0 _Bookworm_ - Official amd64 DVD Bina
ry-1 wit>
```

- Let's comment out the following line and save the file with the changes:

```
deb http://security.debian.org/debian-security bookworm-security main con
trib n>
deb-src http://security.debian.org/debian-security bookworm-security main
contr>
```

- If you wish, you can remove everything from the file by inserting only the following repositories:

```
deb http://deb.debian.org/debian/ bookworm-updates main contrib non-free
non-fr>
deb-src http://deb.debian.org/debian/ bookworm-updates main contrib non-f
ree no>
```

We should replace the list of repositories in our system only when it is really necessary, in cases where incorrect information appears in the list of repositories or the file with the list itself is damaged.

2. Update the database of updated repositories.

```
~$ apt update
```

3. You may need to install `cut` , `nano` , `wget` (but usually the package is already installed):

```
~$ apt install cut nano wget
```

4. Adding 32-bit architecture (for 64-bit systems):

```
~$ sudo dpkg --add-architecture i386
```

5. Set up sudo for yourself as root :

- open the file `sudoers`

```
~$ nano /etc/sudoers
```

- add `'user'` your username instead:

```
# User privilege specification
 root    ALL=(ALL:ALL) ALL
'user' ALL=(ALL:ALL) ALL
```

- go to the directory `lightdm.conf.d`

```
~$ cd /usr/share/lightdm/lightdm.conf.d
```

- open the file `01_debian.conf`

```
 ~$ nano 01_debian.conf
```

- change it `true` to `false` :

```
[Seat:*]
greeter-session=lightdm-greeter
greeter-hide-users=false
session-wrapper=/etc/X11/Xsession
```

- then, after saving, install a tool for configuring and managing the appearance and behavior of LightDM GTK+ Greeter, which is a graphical interface for logging into the LightDM system with the command:

5. Set up sudo for yourself as root :

```
~$ cd /usr/share/lightdm/lightdm.conf.d
```

- open the file `sudoers`

- open the file `01_debian.conf`

```
~$ nano /etc/sudoers
```

```
 ~$ nano 01_debian.conf
```

- add `'user'` your username instead:

- change it `true` to `false` :

```
# User privilege specification
 root    ALL=(ALL:ALL) ALL
'user' ALL=(ALL:ALL) ALL
[Seat:*]
greeter-session=lightdm-greeter
greeter-hide-users=false
```

- go to the directory `lightdm.conf.d`

```
session-wrapper=/etc/X11/Xsession
```

- then, after saving, install a tool for configuring and managing the appearance and behavior of LightDM GTK+ Greeter, which is a graphical interface for logging into the LightDM system with the command:

```
~$ apt install lightdm-gtk-greeter-settings
```

6. Configure automatic login when the system starts.

To do this you need to edit the file `lightdm.conf` :

```
~$ nano /etc/lightdm/lightdm.conf
```

In the window that opens for editing `lightdm.conf` , after the **[Seat:*]** parameter we find the lines:

```
#autologin-user=
#autologin-user-timeout=0
```

You need to uncomment both lines and enter your username in the `autologin-user= ...` for which automatic login is scheduled.

Next, don't forget to save the corrected configuration file.

9. Add the user to the sudo group (sudo activation) with the commands:

```
#$Пример:

autologin-user=my_name
autologin-user-timeout=0
```

```
~$ adduser имя_пользователя sudo
```

7. Install packages for Bluetooth:

10. Reboot the computer

```
~$ sudo apt install blueman blueman bluez pulseaudio-module-bluetooth
```

```
~$ reboot
```

8. Synchronize the GTK theme with Qt :

```
~$  apt install qt5ct qt5-style-plugins
```

automatic login is scheduled.

Next, don't forget to save the corrected configuration file.

9. Add the user to the sudo group (sudo activation) with the commands:

```
#Пример:
~$ su -

autologin-user=my_name
autologin-user-timeout=0
```

```
~$ adduser имя_пользователя sudo
```

7. Install packages for Bluetooth:

10. Reboot the computer

```
~$ sudo apt install blueman blueman bluez pulseaudio-module-bluetooth
```

```
~$ reboot
```

8. Synchronize the GTK theme with Qt :

```
~$  apt install qt5ct qt5-style-plugins
```

After reboot:

11. I update the computer system using sudo and the previously connected 32-bit architecture:

```
~$ sudo apt update && sudo apt upgrade -y
```

12. Installing Python3 on the system:

In theory, Linux distributions can have multiple versions of Python installed, but there can only be one version by default. Setting up Python 3.11 by default requires some additional steps. Follow along.

- **Let's run the ls** command to see what Python binaries are available on your system:

```
~$ ls /usr/bin/python*
```

- **If the version suits us, then we install the full Python3 package with the command:**

```
 /usr/bin/python3     /usr/bin/python3.11-config

 /usr/bin/python3.11  /usr/bin/python3-config
```

```
~$ sudo apt install python3-full
```

- **Let's check** the pre-installed version of *Python3* on our system with the command:
- **Install** the pip package manager **:**

```
~$ python3 --version
~$ sudo apt install python3-pip
```

Python 3.11.5

- **We check** its installation with the command:
- Add repository:

```
~$ pip -V
```

```
~$ sudo add-apt-repository ppa:deadsnakes/ppa
```

pip 23.0.1 from /usr/lib/python3/dist-packages/pip (python 3.11)

```
~$ ls /usr/bin/python*
```

- **If the version suits us, then we install the full Python3 package with the command:**

```
 /usr/bin/python3     /usr/bin/python3.11-config

 /usr/bin/python3.11  /usr/bin/python3-config
```

```
~$ sudo apt install python3-full
```

- **Let's check** the pre-installed version of *Python3* on our system with the command:
- **Install** the pip package manager **:**

```
~$ python3 --version
~$ sudo apt install python3-pip
```

Python 3.11.5

- **We check** its installation with the command:
- Add repository:

```
~$ pip -V
```

```
~$ sudo add-apt-repository ppa:deadsnakes/ppa
```

pip 23.0.1 from /usr/lib/python3/dist-packages/pip (python 3.11)

- **If the version of the pre-installed Python does not suit you** ,

  then before you do anything, make sure that you know which applications depend on the Python3 version already pre-installed on the system, which you received as a result of executing the command in the terminal:

```
~$ python3 --version
```

```
Python 3.11.5
```

- Let's look at the dependencies using the command:

```
~$ apt-cache rdepends python3.11
```

```
python3.11
Reverse Depends:
  libpython3.11-testsuite
  python3-uno
  virtnbdbackup
  stimfit
  python3-stfio
  python3-skorch
  rhythmbox-plugins
  python3-torchvision
  python3-torchtext
  python3-torchaudio
  python3.11-venv
  python3.11-minimal
  python3.11-full
  python3.11-doc
  python3.11-dev
  python3.11-dbg
  python3
  idle-python3.11
  idle-python3.11
  python3-all
  cluster-glue
  python3-escript-mpi
```

```
    python3.11
    Reverse Depends:
      libpython3.11-testsuite
      python3-uno
      virtnbdbackup
      stimfit
      python3-stfio
      python3-skorch
      rhythmbox-plugins
      python3-torchvision
      python3-torchtext
      python3-torchaudio
      python3.11-venv
      python3.11-minimal
      python3.11-full
      python3.11-doc
      python3.11-dev
      python3.11-dbg
      python3
      idle-python3.11
      idle-python3.11
      python3-all
      cluster-glue
      python3-escript-mpi
      python3-escript
      plasma-firewall
      pitivi
      obs-studio
      liferea
      python3-sbml5
      python3-uno
      atac
      kitty
      kdevelop-python
      libglib2.0-tests
      gedit
```

- **Install** the new version of python 3.12:

```
~$ sudo apt install python3.12-full
```

- **Adding** Python 3.11 and 3.12 to **Update Alternatives** :

  To add both versions of Python to the "update-alternatives" utility, run the following commands:

```
~$ sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/p
ython3.11 2
~$ sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/p
ython3.12 1
```

Here:

`/usr/bin/python3` is a symbolic link to the current version of Python 3.

`/usr/bin/python3.11` and `/usr/bin/python3.12` are the paths to the installed versions of Python 3.11 and 3.12, respectively.

`1` and `2` are priorities, where higher priority means a more preferred version.

- **Let's check** the successful installation of the default version of Python3 on our system with the command:

```
~$ python3 -V
```

Python 3.12.2

- **Selecting** the new default Python version:

Now, after running the above commands, we will be able to choose which version of Python 3 to use by running the command:

```
~$ sudo update-alternatives --config python3
```

```
    There is 1 choice for the alternative python3 (providing /usr/bin/pyt
hon3).

      Selection    Path                 Priority   Status
    ------------------------------------------------------------
    * 0            /usr/bin/python3.12   1         auto mode
      1            /usr/bin/python3.11   2        manual mode
      2            /usr/bin/python3.12   1         manual mode

    Press <enter> to keep the current choice[*], or type selection numbe
r:
```

```
~$ sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.11 2
~$ sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.12 1
```

Here:

`/usr/bin/python3` is a symbolic link to the current version of Python 3.

`/usr/bin/python3.11` and `/usr/bin/python3.12` are the paths to the installed versions of Python 3.11 and 3.12, respectively.

`1` and `2` are priorities, where higher priority means a more preferred version.

- **Let's check** the successful installation of the default version of Python3 on our system with the command:

```
~$ python3 -V
```

Python 3.12.2

- **Selecting** the new default Python version:

Now, after running the above commands, we will be able to choose which version of Python 3 to use by running the command:

```
~$ sudo update-alternatives --config python3
```

```
There is 1 choice for the alternative python3 (providing /usr/bin/python3).

  Selection    Path                 Priority   Status
------------------------------------------------------------
* 0            /usr/bin/python3.12   1         auto mode
  1            /usr/bin/python3.11   2        manual mode
  2            /usr/bin/python3.12   1         manual mode

Press <enter> to keep the current choice[*], or type selection number:
```

Important:

*After completing these steps, we can switch between **Python 3.11** and **Python 3.12** using the **"update-alternatives"** utility depending on our needs.*

13. With one command I install the utilities I need

```
~$ sudo apt install gdebi ntfs-3g gtkhash thunar-gtkhash nautilus fuseiso gnome-disk-utility gnome-system-tools synaptic firmware-misc-nonfree curl apt-transport-https dirmngr
   ttf-mscorefonts-installer fonts-freefont-otf fonts-freefont-ttf fonts-noto-core rar unrar libavcodec-extra
```

Here are explanations of the utilities and packages listed above:

Utilities	
ntfs-3g	This is a driver for reading and writing NTFS file systems, which are often used in Windows operating systems. It allows Linux users to interface with NTFS drives.
gtkhash	This is a tool for calculating and checking file hashes. It helps verify the integrity and authenticity of files by comparing their hash amounts.
thunar-gtkhash	This is a plugin for the Thunar file manager that adds GtkHash functionality to calculate file hashes directly from Thunar.
nautilus	This is a file manager for the GNOME graphical environment. It provides a user-friendly interface for navigating through files and folders.
fuseiso	This utility allows you to mount optical disk images (such as ISO images) as a file system. It allows you to view and work with the contents of images without actually writing them to disk.
gnome-disk-utility	This is a disk management tool in the GNOME environment. It provides information about your hard drive, allows you to manage partitions and perform other disk-related operations.
gnome-system-tools	This set of utilities provides a graphical interface for configuring various system settings in the GNOME environment. It includes tools for managing users, network, and other system settings.

Here are explanations of the utilities and packages listed above:

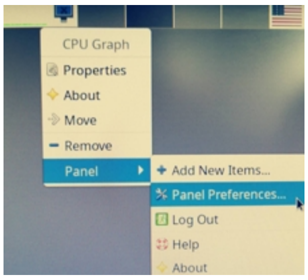
Utilities	
ntfs-3g	This is a driver for reading and writing NTFS file systems, which are often used in Windows operating systems. It allows Linux users to interface with NTFS drives.
gtkhash	This is a tool for calculating and checking file hashes. It helps verify the integrity and authenticity of files by comparing their hash amounts.
thunar-gtkhash	This is a plugin for the Thunar file manager that adds GtkHash functionality to calculate file hashes directly from Thunar.
nautilus	This is a file manager for the GNOME graphical environment. It provides a user-friendly interface for navigating through files and folders.
fuseiso	This utility allows you to mount optical disk images (such as ISO images) as a file system. It allows you to view and work with the contents of images without actually writing them to disk.
gnome-disk-utility	This is a disk management tool in the GNOME environment. It provides information about your hard drive, allows you to manage partitions and perform other disk-related operations.
gnome-system-tools	This set of utilities provides a graphical interface for configuring various system settings in the GNOME environment. It includes tools for managing users, network, and other system settings.
synaptic	This is a graphical interface for managing packages on Debian-like systems. It makes it easy to install, update, and remove programs and packages.
firmware-misc-nonfree	This package contains non-free (proprietary) drivers and firmware for various devices. It can be useful if you need additional drivers to work with your hardware.
curl	Command line utility for making HTTP requests. It is used for downloading files from the Internet, sending data to servers and other network operations.
apt-transport-https	This package adds HTTPS support for APT (Advanced Package Tool), allowing secure downloading of packages from repositories.
dirmngr	A utility for managing GnuPG keys used to verify package and repository signatures.
ttf-mscorefonts-installer, fonts-freefont-otf и fonts-freefont-ttf	This package installs Microsoft Core Fonts such as Arial and Times New Roman on your system.
fonts-noto-core	This package contains the Noto font family developed by Google and provides support for multiple languages and character sets.
rar and unrar	Utilities for creating and extracting files in the RAR format, which is one of the popular data compression formats.
libavcodec-extra	This package contains additional libraries for encoding and decoding audio and video files. It may be required to support some media formats.

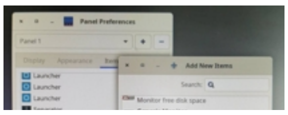
How to add a layout indicator to a panel in XFCE?

If after installation or for some other reason (accidentally deleted via *autoremove*) there is no keyboard layout indicator on the panel, then this problem can be solved in a simple way:

```
~$ sudo apt install xfce4-xkb-plugin
```

Then right-click on the panel. Item "Panel" → "Add new elements", look for the item "Keyboard Layouts", select it and press the "Add" button. A checkbox appears. If you want, right-click on the flag, select "Move" and move it anywhere. Close the "Adding new elements" window.



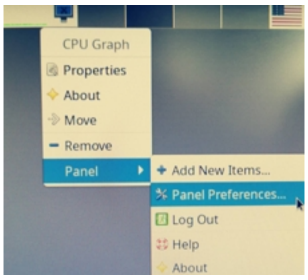


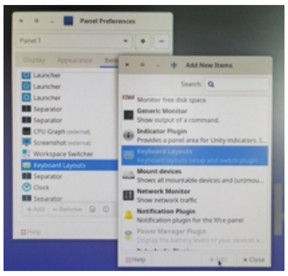
How to add a layout indicator to a panel in XFCE?

If after installation or for some other reason (accidentally deleted via *autoremove*) there is no keyboard layout indicator on the panel, then this problem can be solved in a simple way:

```
~$ sudo apt install xfce4-xkb-plugin
```

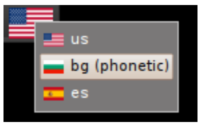
Then right-click on the panel. Item "Panel" → "Add new elements", look for the item "Keyboard Layouts", select it and press the "Add" button. A checkbox appears. If you want, right-click on the flag, select "Move" and move it anywhere. Close the "Adding new elements" window.





xfce4-xkb-plugin is a plugin for managing multiple keyboard layouts

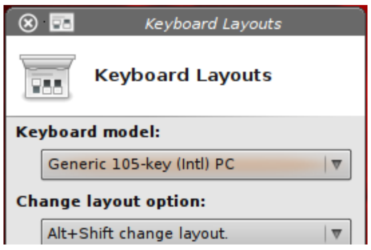
It allows you to select the keyboard model, the key combination for switching between layouts, the actual keyboard layouts, how the current layout is displayed (country flag image or text) and the layout policy, which is whether to save the layout globally (for all windows), per application or for each window.



1. Plugin installation:

```
~$ sudo apt install xfce4-xkb-plugin
```

1. Language selection:



Keyboard Layouts

Keyboard Layouts

Keyboard model:

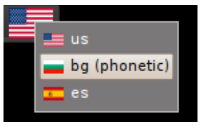
Generic 105-key (Intl) PC

Change layout option:

Alt+Shift change layout.

xfce4-xkb-plugin is a plugin for managing multiple keyboard layouts

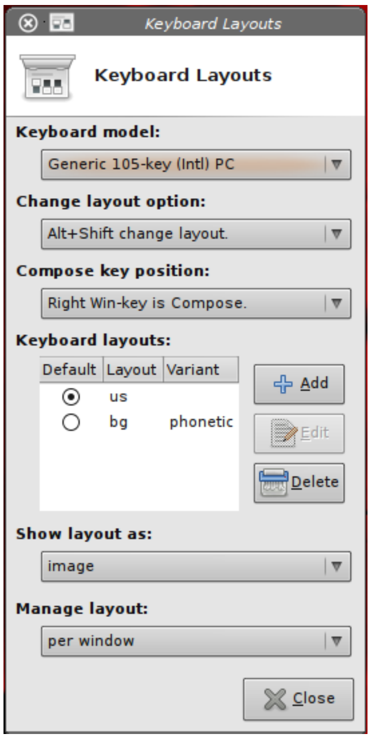
It allows you to select the keyboard model, the key combination for switching between layouts, the actual keyboard layouts, how the current layout is displayed (country flag image or text) and the layout policy, which is whether to save the layout globally (for all windows), per application or for each window.



1. Plugin installation:

```
~$ sudo apt install xfce4-xkb-plugin
```

1. Language selection:



Keyboard Layouts

Keyboard model: Generic 105-key (Intl) PC

Change layout option: Alt+Shift change layout.

Compose key position: Right Win-key is Compose.

Keyboard layouts:

Default	Layout	Variant
●	us	
○	bg	phonetic

Show layout as: image

Manage layout: per window

14. Installing kernel headers and kernel modules

This setup can be useful if you need to compile and install additional drivers or programs that depend on the Linux kernel.

If you do not plan to compile additional kernel modules or install drivers not provided by standard Debian tools, then installing kernel headers and modules may not be necessary.

```
~$ sudo apt install wireless-tools
```

```
~$ sudo apt install linux-headers-$(uname -r|sed 's,[^-]*-[^-]*-,,')
```

17. I will install a graphical interface for managing network connections and disks

15. Installing additional binaries (proprietary binaries)

```
~$ sudo apt install network-manager-gnome gnome-disk-utility
```

For some wireless adapters and devices, they may be required to fully support certain hardware components on your computer. These binaries include firmware for some wireless cards and other devices.

18. Reboot!!!

```
~$ sudo apt install firmware-linux-nonfree
```

```
~$ sudo reboot
```

16. Installing additional tools for managing wireless networks

These tools can be useful for performing various tasks related to configuring and monitoring wireless connections. For example, with their help you can scan available wireless networks, configure connection settings, manage wireless interfaces, and much more.

Linux kernel.

If you do not plan to compile additional kernel modules or install drivers not provided by standard Debian tools, then installing kernel headers and modules may not be necessary.

```
~$ sudo apt install wireless-tools
```

```
~$ sudo apt install linux-headers-$(uname -r|sed 's/[^-]*-[^-]*-//')
```

17. I will install a graphical interface for managing network connections and disks

15. Installing additional binaries (proprietary binaries)

```
~$ sudo apt install network-manager-gnome gnome-disk-utility
```

For some wireless adapters and devices, they may be required to fully support certain hardware components on your computer. These binaries include firmware for some wireless cards and other devices.

18. Reboot!!!

```
~$ sudo apt install firmware-linux-nonfree
```

```
~$ sudo reboot
```

16. Installing additional tools for managing wireless networks

These tools can be useful for performing various tasks related to configuring and monitoring wireless connections. For example, with their help you can scan available wireless networks, configure connection settings, manage wireless interfaces, and much more.

After reboot, Xfce setup :

> After rebooting **Debian 12 "Bookworm" with the Xfce** workspace , I customize it to my needs.
>
> **Xfce** is easy on the system and offers a lot of customization and options to personalize my workspace.

19. Installing Xfce utilities with one command.

Each of these components provides additional functionality and customization options to the Xfce desktop environment, making it more flexible and user-friendly.

You can install or disable them depending on your needs.

```
~$ sudo apt install xfce4-battery-plugin xfce4-clipman xfce4-clipman-plug
in xfce4-cpufreq-plugin xfce4-datetime-plugin xfce4-diskperf-plugin xfce4
-fsguard-plugin
   xfce4-genmon-plugin xfce4-goodies xfce4-mount-plugin xfce4-sensors-plu
gin xfce4-smartbookmark-plugin xfce4-timer-plugin xfce4-wavelan-plugin xf
ce4-power-manager-plugins
```

Here is a brief description of each of the listed plugins and utilities for the Xfce desktop environment :

Utilities	
xfce4-battery-plugin	This plugin is designed to display information about the laptop battery status on a panel. It shows the charge level, battery life and other battery information.
xfce4-clipman и xfce4-clipman-plugin	These components provide a clipboard manager for Xfce. They allow you to copy and paste text or other data between different applications
xfce4-cpufreq-plugin	This plugin allows you to monitor and manage CPU frequency scaling on your computer. It can be useful for optimizing performance and managing power consumption.
xfce4-datetime-plugin	This plugin displays the current time and date on the panel. It can be configured to display different date and time formats.
xfce4-diskperf-plugin	This plugin is designed to monitor disk performance and display information about the read/write speed of data on the disk.
xfce4-fsguard-plugin	This plugin provides free disk space monitoring and notifies the user if free space is running low.
xfce4-genmon-plugin	This plugin allows you to create custom scripts (scripts) and display the result of their execution on the panel. It is useful for monitoring various system parameters.
xfce4-goodies	This package includes a set of various plugins and utilities for the Xfce environment, including the ones discussed above and other useful tools.

```
    xfce4-genmon-plugin xfce4-goodies xfce4-mount-plugin xfce4-sensors-plu
gin xfce4-smartbookmark-plugin xfce4-timer-plugin xfce4-wavelan-plugin xf
ce4-power-manager-plugins
```

Here is a brief description of each of the listed plugins and utilities for the Xfce desktop environment :

Utilities	
xfce4-battery-plugin	This plugin is designed to display information about the laptop battery status on a panel. It shows the charge level, battery life and other battery information.
xfce4-clipman и xfce4-clipman-plugin	These components provide a clipboard manager for Xfce. They allow you to copy and paste text or other data between different applications
xfce4-cpufreq-plugin	This plugin allows you to monitor and manage CPU frequency scaling on your computer. It can be useful for optimizing performance and managing power consumption.
xfce4-datetime-plugin	This plugin displays the current time and date on the panel. It can be configured to display different date and time formats.
xfce4-diskperf-plugin	This plugin is designed to monitor disk performance and display information about the read/write speed of data on the disk.
xfce4-fsguard-plugin	This plugin provides free disk space monitoring and notifies the user if free space is running low.
xfce4-genmon-plugin	This plugin allows you to create custom scripts (scripts) and display the result of their execution on the panel. It is useful for monitoring various system parameters.
xfce4-goodies	This package includes a set of various plugins and utilities for the Xfce environment, including the ones discussed above and other useful tools.
xfce4-mount-plugin	This plugin provides quick access to tools for mounting and unmounting various devices, such as USB drives and network drives.
xfce4-sensors-plugin	This plugin allows you to monitor information about temperature sensors, voltage sensors and other system parameters on your computer.
xfce4-smartbookmark-plugin	This plugin is designed to manage bookmarks in the Thunar file manager.
xfce4-timer-plugin	This plugin allows you to set timers and count down time on the panel.
xfce4-wavelan-plugin	This plugin is designed to monitor wireless networks (Wi-Fi) and display information about networks on the panel.
xfce4-power-manager-plugins	This package includes additional plugins for power management and power saving settings on your computer.

20. Installing utilities for configuring and managing the firewall: ufw and gufw

ufw (Uncomplicated Firewall) is a text-based interface for configuring a firewall on the command line. **ufw** simplifies firewall management by providing an easy way to add rules to allow or block network traffic.

gufw (Graphical Uncomplicated Firewall) is a GUI for **ufw** that makes firewall configuration more intuitive and accessible for desktop environment users.

21. Install the utility for installing .deb packages

```
~$ sudo apt install ufw gufw
```

GDebi is used to install .deb packages using a graphical interface. You can simply double-click on **the .deb file** and **GDebi** will open it and offer to install the package, handling all dependencies automatically.

As an alternative, you can download my ready-made script from the directory `setup-ufw.sh` and run it like this:

```
~$ sudo apt install gdebi -y
```

- Save this setup-ufw.sh file to your computer, then make it executable using the command:

  ~$ chmod +x setup-ufw.sh

22. Themes and icons (icons)

- You can now run this script to configure ufw by running it as root:

  ~$ sudo ./setup-ufw.sh

The script `setup-ufw.sh` will execute all the commands one by one, configure the firewall and enable it every time the system boots. Make sure you have administrative rights (sudo) to run these commands.

One of the first things I did was choose a theme and icons. **Xfce** offers a variety of themes and icons, and I've chosen the ones I like.

Everyone can select and install the theme and icons they like from the resource www.xfce-look.org: -> Themes (https://www.xfce-look.org/browse?cat=138&ord=latest)

All themes and icons are installed respectively in the folders of the home directory:

- **~/.themes** - for the theme
- **~/.icons** - for icons

21. Install the utility for installing .deb packages

```
~$ sudo apt install ufw gufw
```

GDebi is used to install .deb packages using a graphical interface. You can simply double-click on **the .deb file** and **GDebi** will open it and offer to install the package, handling all dependencies automatically.

As an alternative, you can download my ready-made script from the directory `setup-ufw.sh` and run it like this:

```
~$ sudo apt install gdebi -y
```

- Save this setup-ufw.sh file to your computer, then make it executable using the command:

~$ chmod +x setup-ufw.sh

22. Themes and icons (icons)

- You can now run this script to configure ufw by running it as root:

One of the first things I did was choose a theme and icons. **Xfce** offers a variety of themes and icons, and I've chosen the ones I like.

~$ sudo ./setup-ufw.sh

Everyone can select and install the theme and icons they like from the resource www.xfce-look.org: -> Themes (https://www.xfce-look.org/browse?cat=138&ord=latest)

The script `setup-ufw.sh` will execute all the commands one by one, configure the firewall and enable it every time the system boots. Make sure you have administrative rights (sudo) to run these commands.

All themes and icons are installed respectively in the folders of the home directory:

- **~/.themes** - for the theme
- **~/.icons** - for icons

Alternatively, you can install in the directories:

- `/usr/share/icons/` - usually contains preset icons (common to all users)
- `/usr/share/themes` - there are design themes (common for all users)

Now my system looks stylish and modern.

Next, I customized the taskbars and menus. I've added a few shortcuts to the taskbar for quick access to my favorite apps. I also customized the menu to only show apps that I use frequently.

Now my system looks stylish and modern. Taskbars and menus

Next, I customized the taskbars and menus. I've added a few shortcuts to the taskbar for quick access to my favorite apps. I also customized the menu so that it only shows the apps I use frequently. I configured hotkeys to quickly launch apps and perform other frequently used commands.

Now I can easily control my system using hotkeys.

```
~$ ./install_vscode.sh
```

VS Code

The script will take all the necessary steps to add the **Visual Studio Code** repository , install dependencies, and install the application itself on your **Debian** system .

23. Installation of MS Visual Studio Code for Debian

- Copy the **install_vscode.sh** script from the repository to your computer.

- Make it executable with the command:

```
~$ chmod +x install_vs
```

- Then run the script:

git

```
~$ ./install_vscode.sh
```

VS Code

The script will take all the necessary steps to add the **Visual Studio Code** repository , install dependencies, and install the application itself on your **Debian** system .

23. Installation of MS Visual Studio Code for Debian

- Copy the **install_vscode.sh** script from the repository to your computer.

- Make it executable with the command:

```
~$ chmod +x install_vs
```

- Then run the script:

git

24. Installation from source!

I always install `git` from source because it installs the latest version **of Git** with all the components, making it the most complete and convenient way to install and use Git on Debian.

1. To install all dependencies used for the upcoming build and installation of Git binaries, you must:

```
~$ sudo apt-get install dh-autoreconf libcurl4-gnutls-dev libexpat1-dev
\gettext libz-dev libssl-dev
```

2. In order to collect documentation in various *doc, html, info* formats, I will install additional dependencies:

```
~$ sudo apt-get install asciidoc xmlto docbook2x
```

3. Install the *install-info* package:

```
~$ tar -xvf git-2.42.0.tar.xz
```

```
~$ sudo apt-get install install-info
```

```
~$ cd git-2.42.0
```

4. Download the latest version **of Git** , you can download the latest source archive from the following places:

```
~$ make configure
```

- with the [Kernel.org site (https://www.kernel.org/pub/software/scm/git)](https://www.kernel.org/pub/software/scm/git)
- from the mirror on [GitHub (https://github.com/git/git/releases)](https://github.com/git/git/releases)
- Then run the command in the terminal specifying your version (in my case it is *git-2.42.0.tar.xz*):

```
~$ ./configure --prefix=/usr
```

```
 ~$ wget https://mirrors.edge.kernel.org/pub/software/scm/git/git-2.42.0.
tar.xz
```

```
~$ make all doc info
```

```
~$ sudo make install install-doc install-html install-info
```

5. Then I compile and install Git, executing the commands in the following order:

3. Install the *install-info* package :

```
~$ sudo apt-get install install-info
```

4. Download the latest version **of Git** , you can download the latest source archive from the following places:

- with the [Kernel.org site (https://www.kernel.org/pub/software/scm/git)](https://www.kernel.org/pub/software/scm/git)
- from the mirror on [GitHub (https://github.com/git/git/releases)](https://github.com/git/git/releases)
- Then run the command in the terminal specifying your version (in my case it is *git-2.42.0.tar.xz*):

```
 ~$ wget https://mirrors.edge.kernel.org/pub/software/scm/git/git-2.42.0.
tar.xz
```

5. Then I compile and install Git, executing the commands in the following order:

```
~$ tar -xvf git-2.42.0.tar.xz

~$ cd git-2.42.0

~$ make configure

~$ ./configure --prefix=/usr

~$ make all doc info

~$ sudo make install install-doc install-html install-info
```

6. Test the installation and add your entries to the Git configuration file:

```
~$ sudo nano ~/.gitconfig
```

An example of the contents of the configuration file can be found [here (https://gist.github.com/pksunkara/988716)](https://gist.github.com/pksunkara/988716)

Or you can use simpler **.gitconfig entries:**

```
git config --global user.name "Simona Igls"
git config --global user.email simona@igls.io
```

- I install **Visual Studio Code** as a **Git** editor :

```
git config --global core.editor "code --wait"
```

I'll add it to my **.gitignore** file using the

```
# Ignore VSCode Works
*.code-workspace
```

25. Installing JupyterLab Desktop

For my work, I often use **JupyterLab Desktop** because it is a powerful integrated development tool for data analysis and scientific computing. **JupyterLab** provides a convenient environment for creating and running notebooks in which I can combine code, text, graphics, and computational results in a single document.

JupyterLab also provides a wide selection of programming languages, tools, and libraries, making it an ideal tool for data science, machine learning, and research. It has flexible configuration and support for extensions, which allows me to adapt the development environment to my needs.

```
git config --global core.editor "code --wait"
```

I'll add it to my **.gitignore** file using the

```
# Ignore VSCode Works
*.code-workspace
```

25. Installing JupyterLab Desktop

For my work, I often use **JupyterLab Desktop** because it is a powerful integrated development tool for data analysis and scientific computing. **JupyterLab** provides a convenient environment for creating and running notebooks in which I can combine code, text, graphics, and computational results in a single document.

JupyterLab also provides a wide selection of programming languages, tools, and libraries, making it an ideal tool for data science, machine learning, and research. It has flexible configuration and support for extensions, which allows me to adapt the development environment to my needs.

I wrote a script `install_jupyterlab_desktop.sh` to install **JupyterLab Desktop** :

- Copy the script `install_jupyterlab_desktop.sh` from the repository to your computer.
- Make it executable with the command:

```
~$ chmod +x install_jupyterlab_desktop.sh
```

- Then run the script:

```
~$ ./install_jupyterlab_desktop.sh
```

```
~$ sudo apt install puddletag qsynth simplescreenrecorder seahorse soundc
onverter timidity uget winff engrampa mpg321 vorbis-tools grub-customizer
filezilla isomaster qshutdown
   gparted easytag cherrytree xfce4-screenshooter mtools kcolorchooser on
board [illegible] gstreamer1.0-pulseaudio modem-manager-gui sox libsox-fmt-
al
```

26. Installing additional programs

I install the programs I need in blocks; these blocks can be edited by deleting or adding the programs you need:

- Installing drivers for printers:

```
~$ sudo apt install audacity audacious bleachbit cpufrequtils clamav clam
tk evince fbreader fluid-soundfont-gm fluid-soundfont-gs gnome-mpv gpick
gvidm gpicview guvcview mediainfo
   mediainfo-gui mkvtoolnix usb-modeswitch net-tools gkrellm plank
```

```
~$ sudo apt-get install printer-driver-all
```

```
~$ sudo apt-get install cups hplip
```

- Google Chrome installation:

```
~$ sudo apt install puddletag qsynth simplescreenrecorder seahorse soundc
onverter timidity uget winff engrampa mpg321 vorbis-tools grub-customizer
filezilla isomaster qshutdown
   gparted easytag cherrytree xfce4-screenshooter mtools kcolorchooser on
board joystick gstreamer1.0-pulseaudio modem-manager-gui sox libsox-fmt-
al
```

26. Installing additional programs

I install the programs I need in blocks; these blocks can be edited by deleting or adding the programs you need:

- Installing drivers for printers:

```
~$ sudo apt install audacity audacious bleachbit cpufrequtils clamav clam
tk evince fbreader fluid-soundfont-gm fluid-soundfont-gs gnome-mpv gpick
gvidm gpicview guvcview mediainfo
   mediainfo-gui mkvtoolnix usb-modeswitch net-tools gkrellm plank
```

```
~$ sudo apt-get install printer-driver-all
```

```
~$ sudo apt-get install cups hplip
```

- Google Chrome installation:

```
    ~$ cd tmp
```

```
    ~$ sudo wget https://dl.google.com/linux/direct/google-chrome-stable_
current_amd64.deb
```

```
      ~$ sudo apt install ./google-chrome-stable_current_amd64.deb
```

- You can remove the Firefox browser, but this is optional:

```
    ~$ sudo apt remove --purge xarchiver firefox-esr
    ~$ sudo apt autoremove
```

I reduce the shutdown of problem processes from 1.5 minutes to 10 seconds

- I edit the config file with **root** rights `/etc/systemd/system.conf` :

27. ZSH instead of bash

```
   ~$ sudo nano /etc/systemd/system.conf
```

- I recommend uncommenting and correcting the values in these lines to **10s** :

Kali Linux Terminal

Although the **Xfce** terminal works well, I prefer the look and feel of the **Kali Linux** terminal , so I'll write how I do it.

```
 DefaultTimeoutStartSec=10s
 DefaultTimeoutStopSec=10s
```

Installing ZSH on Debian

The Z shell is a Unix shell that was developed as an extension to **the BASH (Bourne shell)** in the early **90s** .

I reduce the shutdown of problem processes from 1.5 minutes to 10 seconds

- I edit the config file with **root** rights `/etc/systemd/system.conf` :

27. ZSH instead of bash

```
~$ sudo nano /etc/systemd/system.conf
```

- I recommend uncommenting and correcting the values in these lines to **10s** :

Kali Linux Terminal

Although the **Xfce** terminal works well, I prefer the look and feel of the **Kali Linux** terminal , so I'll write how I do it.

```
DefaultTimeoutStartSec=10s
DefaultTimeoutStopSec=10s
```

Installing ZSH on Debian

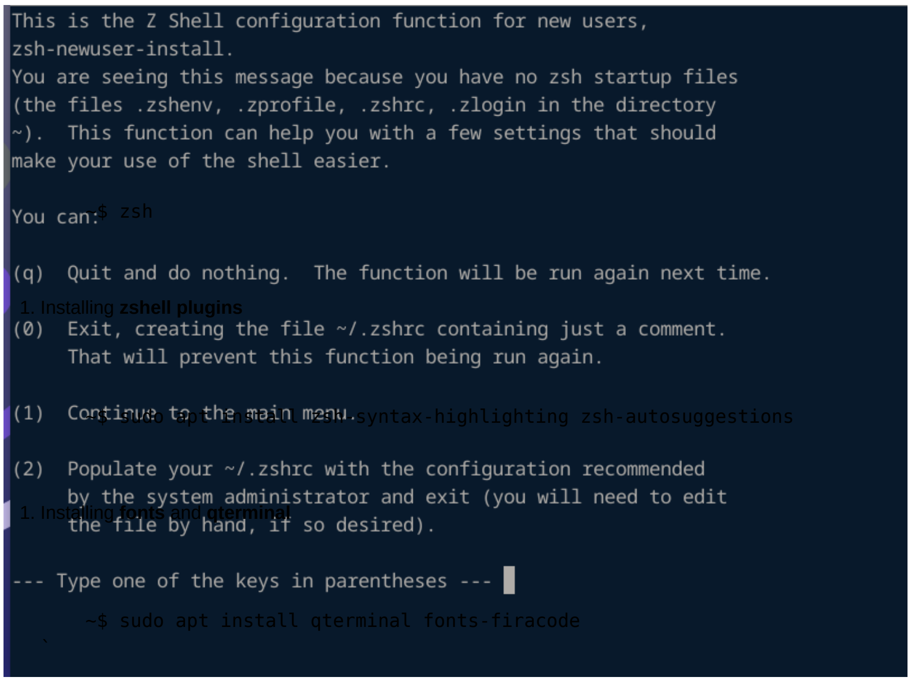
The Z shell is a Unix shell that was developed as an extension to **the BASH (Bourne shell)** in the early **90s** .

Z shell is an interactive shell that includes many features of other **Unix/GNU Linux** shells such as **bash** , **fish** , **dash** and **ksh** .

1. Installing **zshell** :

```
~$ sudo apt update

~$ sudo apt install zsh
```

After installing **zsh** , run the **zsh** command to switch from the **bash** prompt to the **zsh** prompt . **When you run the command, you will see a Z** shell configuration prompt - select **the 0** (zero) option from the prompts and press **enter** to apply:



```
This is the Z Shell configuration function for new users,
zsh-newuser-install.
You are seeing this message because you have no zsh startup files
(the files .zshenv, .zprofile, .zshrc, .zlogin in the directory
~).  This function can help you with a few settings that should
make your use of the shell easier.

You can:

(q)  Quit and do nothing.  The function will be run again next time.

(0)  Exit, creating the file ~/.zshrc containing just a comment.
     That will prevent this function being run again.

(1)  Continue to the main menu.

(2)  Populate your ~/.zshrc with the configuration recommended
     by the system administrator and exit (you will need to edit
     the file by hand, if so desired).

--- Type one of the keys in parentheses ---
```

```
$ zsh
```

1. Installing **zshell plugins**

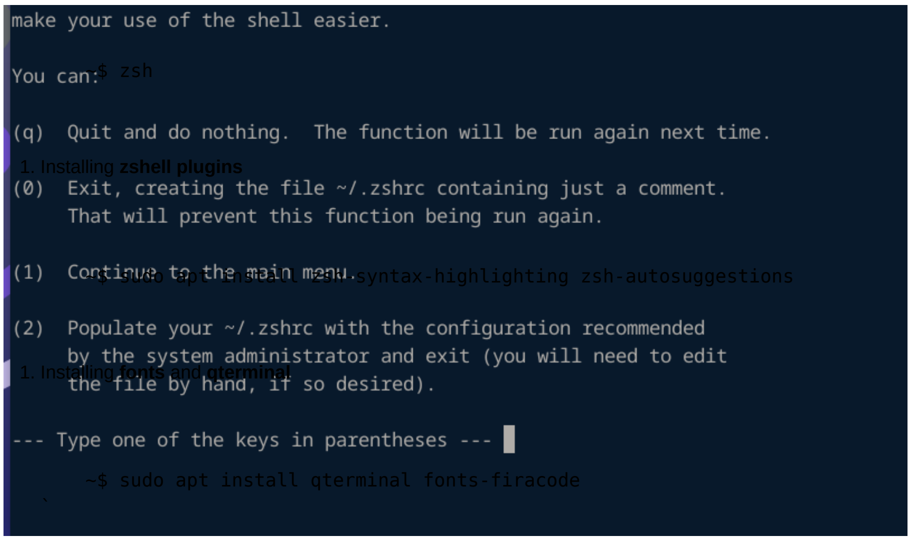
```
~$ sudo apt install zsh-syntax-highlighting zsh-autosuggestions
```

1. Installing **fonts** and **qterminal**

```
~$ sudo apt install qterminal fonts-firacode
```

1. Changing the default login shell

I'll use **chsh** - it's a powerful tool used to change your login shell. There is no need to install the **chsh**



```
make your use of the shell easier.

You can:

(q)  Quit and do nothing.  The function will be run again next time.

(0)  Exit, creating the file ~/.zshrc containing just a comment.
     That will prevent this function being run again.

(1)  Continue to the main menu.

(2)  Populate your ~/.zshrc with the configuration recommended
     by the system administrator and exit (you will need to edit
     the file by hand, if so desired).

--- Type one of the keys in parentheses ---
```

```
$ zsh
```

1. Installing **zshell plugins**

```
~$ sudo apt install zsh-syntax-highlighting zsh-autosuggestions
```

1. Installing **fonts** and **qterminal**

```
~$ sudo apt install qterminal fonts-firacode
`
```

1. Changing the default login shell

I'll use **chsh** - it's a powerful tool used to change your login shell. There is no need to install the **chsh** command as it is a standard package in all Linux distributions.

```
~$ chsh -s /bin/zsh
```

- After you are asked to confirm your password, you must log out and then log in again to see the changes.

***The bash** prompt will be replaced by the **zsh** prompt .*

Setting up zshell in Debian

1. Modifying the **.zshrc file**

The .zshrc file is the startup file equivalent to **the .bashrc** [(download) (https://github.com/Ssobol7/Debian-12-Xfce-My-Config/blob/main/zshrc)](https://github.com/Ssobol7/Debian-12-Xfce-My-Config/blob/main/zshrc) file for **bash** (Bourne Again shell), which is used to configure **zshell** .

1. After successfully creating **the .zshrc** file , I will open it in the **nano text editor**

This file (~/.zshrc) is a hidden file and is located in the home directory.

```
~$ sudo nano ~/.zshrc
```

- and paste the following script into my **.zshrc** file:

- delete the old file

```
~$ rm ~/.zshrc
```

- Create a new .zshrc file

```
~$ touch ~/.zshrc
```

1. After successfully creating **the .zshrc** file , I will open it in the **nano text editor**

This file (~/.zshrc) is a hidden file and is located in the home directory.

```
~$ sudo nano ~/.zshrc
```

and paste the following script into my **.zshrc** file:

- delete the old file

```
~$ rm ~/.zshrc
```

- Create a new .zshrc file

```
~$ touch ~/.zshrc
```

```
# ~/.zshrc file

setopt autocd              # change directory just by typing its name
#setopt correct            # auto correct mistakes
setopt interactivecomments # allow comments in interactive mode
setopt magicequalsubst     # enable filename expansion for arguments of t
he form ‘anything=expression’
setopt nonomatch           # hide error message if there is no match for
the pattern
setopt notify              # report the status of background jobs immedia
tely
setopt numericglobsort     # sort filenames numerically when it makes sen
se
setopt promptsubst         # enable command substitution in prompt

WORDCHARS=${WORDCHARS//\/} # Don't consider certain characters part of th
e word

# hide EOL sign ('%')
PROMPT_EOL_MARK=""

# configure key keybindings
bindkey -e                                        # emacs key bindings
bindkey ' ' magic-space                           # do history expansion
on space
```

```
# ~/.zshrc file

setopt autocd              # change directory just by typing its name
#setopt correct            # auto correct mistakes
setopt interactivecomments # allow comments in interactive mode
setopt magicequalsubst     # enable filename expansion for arguments of t
he form ‘anything=expression’
setopt nonomatch           # hide error message if there is no match for
the pattern
setopt notify              # report the status of background jobs immedia
tely
setopt numericglobsort     # sort filenames numerically when it makes sen
se
setopt promptsubst         # enable command substitution in prompt

WORDCHARS=${WORDCHARS//\/} # Don't consider certain characters part of th
e word

# hide EOL sign ('%')
PROMPT_EOL_MARK=""

# configure key keybindings
bindkey -e                                        # emacs key bindings
bindkey ' ' magic-space                           # do history expansion
on space
bindkey '^U' backward-kill-line                   # ctrl + U
bindkey '^[[3;5~' kill-word                       # ctrl + Supr
bindkey '^[[3~' delete-char                       # delete
bindkey '^[[1;5C' forward-word                    # ctrl + ->
bindkey '^[[1;5D' backward-word                   # ctrl + <-
bindkey '^[[5~' beginning-of-buffer-or-history    # page up
bindkey '^[[6~' end-of-buffer-or-history          # page down
bindkey '^[[H' beginning-of-line                  # home
bindkey '^[[F' end-of-line                        # end
bindkey '^[[Z' undo                               # shift + tab undo last
action

# enable completion features
autoload -Uz compinit
compinit -d ~/.cache/zcompdump
zstyle ':completion:*:*:*:*:*' menu select
zstyle ':completion:*' auto-description 'specify: %d'
zstyle ':completion:*' completer _expand _complete
zstyle ':completion:*' format 'Completing %d'
zstyle ':completion:*' group-name ''
zstyle ':completion:*' list-colors ''
zstyle ':completion:*' list-prompt %SAt %p: Hit TAB for more, or the char
acter to insert%s
zstyle ':completion:*' matcher-list 'm:{a-zA-Z}={A-Za-z}'
zstyle ':completion:*' rehash true
zstyle ':completion:*' select-prompt %SScrolling active: current selectio
n at %p%s
zstyle ':completion:*' use-compctl false
zstyle ':completion:*' verbose true
zstyle ':completion:*:kill:*' command 'ps -u $USER -o pid,%cpu,tty,cputim
e,cmd'

# History configurations
HISTFILE=~/.zsh_history
HISTSIZE=1000
SAVEHIST=2000
setopt hist_expire_dups_first # delete duplicates first when HISTFILE siz
e exceeds HISTSIZE
setopt hist_ignore_dups       # ignore duplicated commands history list
setopt hist_ignore_space      # ignore commands that start with space
setopt hist_verify            # show command with history expansion to us
er before running it
#setopt share_history         # share command history data

# force zsh to show the complete history
alias history="history 0"

# configure `time` format
TIMEFMT=$'\nreal\t%E\nuser\t%U\nsys\t%S\ncpu\t%P'

# make less more friendly for non-text input files, see lesspipe(1)
#[ -x /usr/bin/lesspipe ] && eval "$(SHELL=/bin/sh lesspipe)"
```

```
# set variable identifying the chroot you work in (used in the prompt bel
ow)
if [ -z "${debian_chroot:-}" ] && [ -r /etc/debian_chroot ]; then
    debian_chroot=$(cat /etc/debian_chroot)
fi

# set a fancy prompt (non-color, unless we know we "want" color)
case "$TERM" in
    xterm-color|*-256color) color_prompt=yes;;
esac

# uncomment for a colored prompt, if the terminal has the capability; tur
ned
# off by default to not distract the user: the focus in a terminal window
# should be on the output of commands, not on the prompt
force_color_prompt=yes

if [ -n "$force_color_prompt" ]; then
    if [ -x /usr/bin/tput ] && tput setaf 1 >&/dev/null; then
        # We have color support; assume it's compliant with Ecma-48
        # (ISO/IEC-6429). (Lack of such support is extremely rare, and su
ch
        # a case would tend to support setf rather than setaf.)
        color_prompt=yes
    else
```

```
n at %p%s
zstyle ':completion:*' use-compctl false
zstyle ':completion:*' verbose true
zstyle ':completion:*:kill:*' command 'ps -u $USER -o pid,%cpu,tty,cputim
e,cmd'

# History configurations
HISTFILE=~/.zsh_history
HISTSIZE=1000
SAVEHIST=2000
setopt hist_expire_dups_first # delete duplicates first when HISTFILE siz
e exceeds HISTSIZE
setopt hist_ignore_dups       # ignore duplicated commands history list
setopt hist_ignore_space      # ignore commands that start with space
setopt hist_verify            # show command with history expansion to us
er before running it
#setopt share_history         # share command history data

# force zsh to show the complete history
alias history="history 0"

# configure `time` format
TIMEFMT=$'\nreal\t%E\nuser\t%U\nsys\t%S\ncpu\t%P'

# make less more friendly for non-text input files, see lesspipe(1)
#[ -x /usr/bin/lesspipe ] && eval "$(SHELL=/bin/sh lesspipe)"
```

```
# set variable identifying the chroot you work in (used in the prompt bel
ow)
if [ -z "${debian_chroot:-}" ] && [ -r /etc/debian_chroot ]; then
    debian_chroot=$(cat /etc/debian_chroot)
fi

# set a fancy prompt (non-color, unless we know we "want" color)
case "$TERM" in
    xterm-color|*-256color) color_prompt=yes;;
esac

# uncomment for a colored prompt, if the terminal has the capability; tur
ned
# off by default to not distract the user: the focus in a terminal window
# should be on the output of commands, not on the prompt
force_color_prompt=yes

if [ -n "$force_color_prompt" ]; then
    if [ -x /usr/bin/tput ] && tput setaf 1 >&/dev/null; then
        # We have color support; assume it's compliant with Ecma-48
        # (ISO/IEC-6429). (Lack of such support is extremely rare, and su
ch
        # a case would tend to support setf rather than setaf.)
        color_prompt=yes
    else
        color_prompt=
    fi
fi

configure_prompt() {
    prompt_symbol=㉿
    # Skull emoji for root terminal
    #[ "$EUID" -eq 0 ] && prompt_symbol=💀
    case "$PROMPT_ALTERNATIVE" in
        twoline)
            PROMPT=$'%F{%(#.blue.green)}┌──${debian_chroot:+($debian_chro
ot)─}${VIRTUAL_ENV:+($(basename $VIRTUAL_ENV))─}(%B%F{%(#.red.blue)}%n'$p
rompt_symbol$'%m%b%F{%(#.blue.green)})-[%B%F{reset}%(6~.%-1~/…/%4~.%5~)%
b%F{%(#.blue.green)}]\n└─%B%(#.%F{red}#.%F{blue}$)%b%F{reset} '
            # Right-side prompt with exit codes and background processes
            #RPROMPT=$'%(?.. %? %F{red}%B⨯%b%F{reset})%(1j. %j %F{yello
w}%B⚙%b%F{reset}.)'
            ;;
        oneline)
            PROMPT=$'${debian_chroot:+($debian_chroot)}${VIRTUAL_ENV:+
($(basename $VIRTUAL_ENV))}%B%F{%(#.red.blue)}%n@%m%b%F{reset}:%B%F{%(#.b
lue.green)}%~%b%F{reset}%(#.#.$) '
            RPROMPT=
            ;;
        backtrack)
            PROMPT=$'${debian_chroot:+($debian_chroot)}${VIRTUAL_ENV:+
($(basename $VIRTUAL_ENV))}%B%F{red}%n@%m%b%F{reset}:%B%F{blue}%~%b%F{res
et}%(#.#.$) '
            RPROMPT=
            ;;
    esac
    unset prompt_symbol
}

# The following block is surrounded by two delimiters.
# These delimiters must not be modified. Thanks.
# START KALI CONFIG VARIABLES
PROMPT_ALTERNATIVE=twoline
NEWLINE_BEFORE_PROMPT=yes
# STOP KALI CONFIG VARIABLES

if [ "$color_prompt" = yes ]; then
    # override default virtualenv indicator in prompt
    VIRTUAL_ENV_DISABLE_PROMPT=1

    configure_prompt

    # enable syntax-highlighting
    if [ -f /usr/share/zsh-syntax-highlighting/zsh-syntax-highlighting.zs
h ]; then
        . /usr/share/zsh-syntax-highlighting/zsh-syntax-highlighting.zsh
        ZSH_HIGHLIGHT_HIGHLIGHTERS=(main brackets pattern)
```

```
        ZSH_HIGHLIGHT_STYLES[default]=none
        ZSH_HIGHLIGHT_STYLES[unknown-token]=fg=white,underline
        ZSH_HIGHLIGHT_STYLES[reserved-word]=fg=cyan,bold
        ZSH_HIGHLIGHT_STYLES[suffix-alias]=fg=green,underline
        ZSH_HIGHLIGHT_STYLES[global-alias]=fg=green,bold
        ZSH_HIGHLIGHT_STYLES[precommand]=fg=green,underline
        ZSH_HIGHLIGHT_STYLES[commandseparator]=fg=blue,bold
        ZSH_HIGHLIGHT_STYLES[autodirectory]=fg=green,underline
        ZSH_HIGHLIGHT_STYLES[path]=bold
        ZSH_HIGHLIGHT_STYLES[path_pathseparator]=
        ZSH_HIGHLIGHT_STYLES[path_prefix_pathseparator]=
        ZSH_HIGHLIGHT_STYLES[globbing]=fg=blue,bold
        ZSH_HIGHLIGHT_STYLES[history-expansion]=fg=blue,bold
        ZSH_HIGHLIGHT_STYLES[command-substitution]=none
        ZSH_HIGHLIGHT_STYLES[command-substitution-delimiter]=fg=magenta,b
old
        ZSH_HIGHLIGHT_STYLES[process-substitution]=none
        ZSH_HIGHLIGHT_STYLES[process-substitution-delimiter]=fg=magenta,b
old
        ZSH_HIGHLIGHT_STYLES[single-hyphen-option]=fg=green
        ZSH_HIGHLIGHT_STYLES[double-hyphen-option]=fg=green
        ZSH_HIGHLIGHT_STYLES[back-quoted-argument]=none
        ZSH_HIGHLIGHT_STYLES[back-quoted-argument-delimiter]=fg=blue,bold
        ZSH_HIGHLIGHT_STYLES[single-quoted-argument]=fg=yellow
        ZSH_HIGHLIGHT_STYLES[double-quoted-argument]=fg=yellow
        ZSH_HIGHLIGHT_STYLES[dollar-quoted-argument]=fg=yellow
```

```
($(basename $VIRTUAL_ENV))%B%F{red}%n@%m%b%F{reset}:%B%F{blue}%~%b%F{res
et}%(#.#.$) '
            RPROMPT=
            ;;
    esac
    unset prompt_symbol
}

# The following block is surrounded by two delimiters.
# These delimiters must not be modified. Thanks.
# START KALI CONFIG VARIABLES
PROMPT_ALTERNATIVE=twoline
NEWLINE_BEFORE_PROMPT=yes
# STOP KALI CONFIG VARIABLES

if [ "$color_prompt" = yes ]; then
    # override default virtualenv indicator in prompt
    VIRTUAL_ENV_DISABLE_PROMPT=1

    configure_prompt

    # enable syntax-highlighting
    if [ -f /usr/share/zsh-syntax-highlighting/zsh-syntax-highlighting.zs
h ]; then
        . /usr/share/zsh-syntax-highlighting/zsh-syntax-highlighting.zsh
        ZSH_HIGHLIGHT_HIGHLIGHTERS=(main brackets pattern)
```

```
        ZSH_HIGHLIGHT_STYLES[default]=none
        ZSH_HIGHLIGHT_STYLES[unknown-token]=fg=white,underline
        ZSH_HIGHLIGHT_STYLES[reserved-word]=fg=cyan,bold
        ZSH_HIGHLIGHT_STYLES[suffix-alias]=fg=green,underline
        ZSH_HIGHLIGHT_STYLES[global-alias]=fg=green,bold
        ZSH_HIGHLIGHT_STYLES[precommand]=fg=green,underline
        ZSH_HIGHLIGHT_STYLES[commandseparator]=fg=blue,bold
        ZSH_HIGHLIGHT_STYLES[autodirectory]=fg=green,underline
        ZSH_HIGHLIGHT_STYLES[path]=bold
        ZSH_HIGHLIGHT_STYLES[path_pathseparator]=
        ZSH_HIGHLIGHT_STYLES[path_prefix_pathseparator]=
        ZSH_HIGHLIGHT_STYLES[globbing]=fg=blue,bold
        ZSH_HIGHLIGHT_STYLES[history-expansion]=fg=blue,bold
        ZSH_HIGHLIGHT_STYLES[command-substitution]=none
        ZSH_HIGHLIGHT_STYLES[command-substitution-delimiter]=fg=magenta,b
old
        ZSH_HIGHLIGHT_STYLES[process-substitution]=none
        ZSH_HIGHLIGHT_STYLES[process-substitution-delimiter]=fg=magenta,b
old
        ZSH_HIGHLIGHT_STYLES[single-hyphen-option]=fg=green
        ZSH_HIGHLIGHT_STYLES[double-hyphen-option]=fg=green
        ZSH_HIGHLIGHT_STYLES[back-quoted-argument]=none
        ZSH_HIGHLIGHT_STYLES[back-quoted-argument-delimiter]=fg=blue,bold
        ZSH_HIGHLIGHT_STYLES[single-quoted-argument]=fg=yellow
        ZSH_HIGHLIGHT_STYLES[double-quoted-argument]=fg=yellow
        ZSH_HIGHLIGHT_STYLES[dollar-quoted-argument]=fg=yellow
        ZSH_HIGHLIGHT_STYLES[rc-quote]=fg=magenta
        ZSH_HIGHLIGHT_STYLES[dollar-double-quoted-argument]=fg=magenta,bo
ld
        ZSH_HIGHLIGHT_STYLES[back-double-quoted-argument]=fg=magenta,bold
        ZSH_HIGHLIGHT_STYLES[back-dollar-quoted-argument]=fg=magenta,bold
        ZSH_HIGHLIGHT_STYLES[assign]=none
        ZSH_HIGHLIGHT_STYLES[redirection]=fg=blue,bold
        ZSH_HIGHLIGHT_STYLES[comment]=fg=black,bold
        ZSH_HIGHLIGHT_STYLES[named-fd]=none
        ZSH_HIGHLIGHT_STYLES[numeric-fd]=none
        ZSH_HIGHLIGHT_STYLES[arg0]=fg=cyan
        ZSH_HIGHLIGHT_STYLES[bracket-error]=fg=red,bold
        ZSH_HIGHLIGHT_STYLES[bracket-level-1]=fg=blue,bold
        ZSH_HIGHLIGHT_STYLES[bracket-level-2]=fg=green,bold
        ZSH_HIGHLIGHT_STYLES[bracket-level-3]=fg=magenta,bold
        ZSH_HIGHLIGHT_STYLES[bracket-level-4]=fg=yellow,bold
        ZSH_HIGHLIGHT_STYLES[bracket-level-5]=fg=cyan,bold
        ZSH_HIGHLIGHT_STYLES[cursor-matchingbracket]=standout
    fi
else
    PROMPT='${debian_chroot:+($debian_chroot)}%n@%m:%~%(#.#.$) '
fi
unset color_prompt force_color_prompt

toggle_oneline_prompt(){
    if [ "$PROMPT_ALTERNATIVE" = oneline ]; then
        PROMPT_ALTERNATIVE=twoline
    else
        PROMPT_ALTERNATIVE=oneline
    fi
    configure_prompt
    zle reset-prompt
}
zle -N toggle_oneline_prompt
bindkey ^P toggle_oneline_prompt

# enable color support of ls, less and man, and also add handy aliases
if [ -x /usr/bin/dircolors ]; then
    test -r ~/.dircolors && eval "$(dircolors -b ~/.dircolors)" || eval
"$(dircolors -b)"
    export LS_COLORS="$LS_COLORS:ow=30;44:" # fix ls color for folders wi
th 777 permissions

    alias ls='ls --color=auto'
    #alias dir='dir --color=auto'
    #alias vdir='vdir --color=auto'

    alias grep='grep --color=auto'
    alias fgrep='fgrep --color=auto'
    alias egrep='egrep --color=auto'
    alias diff='diff --color=auto'
    alias ip='ip --color=auto'
```

```
    if [ "$NEWLINE_BEFORE_PROMPT" = yes ]; then
        if [ -z "$_NEW_LINE_BEFORE_PROMPT" ]; then
            _NEW_LINE_BEFORE_PROMPT=1
        else
            print ""
        fi
    fi
}

# If this is an xterm set the title to user@host:dir
case "$TERM" in
xterm*|rxvt*|Eterm|aterm|kterm|gnome*|alacritty)
    TERM_TITLE=$'\e]0;${debian_chroot:+($debian_chroot)}${VIRTUAL_ENV:+
($(basename $VIRTUAL_ENV))}%n@%m: %~\a'
    ;;
*)
    ;;
esac

precmd() {
    # Print the previously configured title
    print -Pnr -- "$TERM_TITLE"

    # Print a new line before the prompt, but only if it is not the first
line
```

```
        PROMPT_ALTERNATIVE=twoline
    else
        PROMPT_ALTERNATIVE=oneline
    fi
    configure_prompt
    zle reset-prompt
}
zle -N toggle_oneline_prompt
bindkey ^P toggle_oneline_prompt

# enable color support of ls, less and man, and also add handy aliases
if [ -x /usr/bin/dircolors ]; then
    test -r ~/.dircolors && eval "$(dircolors -b ~/.dircolors)" || eval
"$(dircolors -b)"
    export LS_COLORS="$LS_COLORS:ow=30;44:" # fix ls color for folders wi
th 777 permissions

    alias ls='ls --color=auto'
    #alias dir='dir --color=auto'
    #alias vdir='vdir --color=auto'

    alias grep='grep --color=auto'
    alias fgrep='fgrep --color=auto'
    alias egrep='egrep --color=auto'
    alias diff='diff --color=auto'
    alias ip='ip --color=auto'

    export LESS_TERMCAP_mb=$'\E[1;31m'     # begin blink
    export LESS_TERMCAP_md=$'\E[1;36m'     # begin bold
    export LESS_TERMCAP_me=$'\E[0m'        # reset bold/blink
    export LESS_TERMCAP_so=$'\E[01;33m'    # begin reverse video
    export LESS_TERMCAP_se=$'\E[0m'        # reset reverse video
    export LESS_TERMCAP_us=$'\E[1;32m'     # begin underline
    export LESS_TERMCAP_ue=$'\E[0m'        # reset underline

    # Take advantage of $LS_COLORS for completion as well
    zstyle ':completion:*' list-colors "${(s.:.)LS_COLORS}"
    zstyle ':completion:*:*:kill:*:processes' list-colors '=(#b) #([0-9]
#)*=0=01;31'
fi

# some more ls aliases
alias ll='ls -l'
alias la='ls -A'
alias l='ls -CF'

# enable auto-suggestions based on the history
if [ -f /usr/share/zsh-autosuggestions/zsh-autosuggestions.zsh ]; then
    . /usr/share/zsh-autosuggestions/zsh-autosuggestions.zsh
    # change suggestion color
    ZSH_AUTOSUGGEST_HIGHLIGHT_STYLE='fg=#999'
fi

# enable command-not-found if installed
if [ -f /etc/zsh_command_not_found ]; then
    . /etc/zsh_command_not_found
fi
compinit
```

```
    if [ "$NEWLINE_BEFORE_PROMPT" = yes ]; then
        if [ -z "$_NEW_LINE_BEFORE_PROMPT" ]; then
            _NEW_LINE_BEFORE_PROMPT=1
        else
            print ""
        fi
    fi
}

# If this is an xterm set the title to user@host:dir
case "$TERM" in
xterm*|rxvt*|Eterm|aterm|kterm|gnome*|alacritty)
    TERM_TITLE=$'\e]0;${debian_chroot:+($debian_chroot)}${VIRTUAL_ENV:+
($(basename $VIRTUAL_ENV))}%n@%m: %~\a'
    ;;
*)
    ;;
esac

precmd() {
    # Print the previously configured title
    print -Pnr -- "$TERM_TITLE"

    # Print a new line before the prompt, but only if it is not the first
line
```

1. First, let's clone the **Kali Linux** color schemes and themes from the **GitHub** repository with the command:

```
$ git clone https://github.com/linuxopsys/ubuntu-to-kali-terminal.git
```

- *DON'T FORGET TO SAVE my .zshrc!!!*

1. After cloning, I go to the **ubuntu-to-kali-terminal** directory and extract the compressed files:
1. For the changes to take effect, I simply close and reopen my terminal.

```
~$ cd ubuntu-to-kali-terminal
~$ tar -xvf color-schemes.tar
~$ tar -xvf kali-dark-theme.tar
```

Downloading Kali Linux Color Schemes and Themes

1. **I delete the qtermwidget5** directory located in the **/usr/share** directory and replace it with the directory from the extracted tar archive:

```
# enable command-not-found if installed
if [ -f /etc/zsh_command_not_found ]; then
    . /etc/zsh_command_not_found
fi
compinit
```

1. First, let's clone the **Kali Linux** color schemes and themes from the **GitHub** repository with the command:

```
$ git clone https://github.com/linuxopsys/ubuntu-to-kali-terminal.git
```

- *DON'T FORGET TO SAVE my .zshrc!!!*

1. After cloning, I go to the **ubuntu-to-kali-terminal** directory and extract the compressed files:
1. For the changes to take effect, I simply close and reopen my terminal.

```
~$ cd ubuntu-to-kali-terminal
~$ tar -xvf color-schemes.tar
~$ tar -xvf kali-dark-theme.tar
```

Downloading Kali Linux Color Schemes and Themes

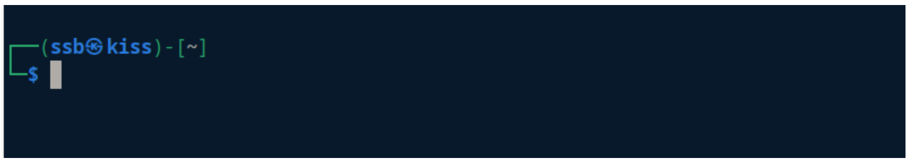
1. **I delete the qtermwidget5** directory located in the **/usr/share** directory and replace it with the directory from the extracted tar archive:

```
sudo rm -rf /usr/share/qtermwidget5
sudo mv -f usr/share/qtermwidget5 /usr/share
```

Changing Qterminal settings

1. Open Qterminal settings: change the color scheme to Kali-Dark and then click the [Apply] button, this will change the terminal theme to Kali.
2. At the bottom of our terminal's appearance settings, I change "Application Transparency" from "0%" to "5%", just like in Kali, and apply it for the changes to take effect.

After making the changes, your **qterminal** should look like this:



```
┌──(ssb㉿kiss)-[~]
└─$ 
```

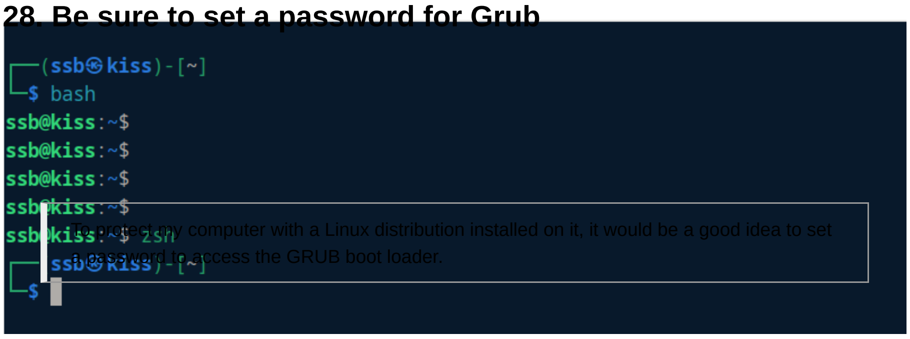
- if you need to temporarily switch to the **bash** shell and then return to the **zsh** shell then run the following commands in a terminal
- to go:

```
~$ bash
```

- for return:

```
~$ zsh
```

28. Be sure to set a password for Grub



```
┌──(ssb㉿kiss)-[~]
└─$ bash
ssb@kiss:~$
ssb@kiss:~$
ssb@kiss:~$
ssb@kiss:~$
ssb@kiss:~$ zsh
┌──(ssb㉿kiss)-[~]
└─$ 
```

To protect my computer with a Linux distribution installed on it, it would be a good idea to set a password to access the GRUB boot loader.

This is done in three steps:

1) Generate a password hash

- to go:

```
~$ bash
```

- for return:

```
~$ zsh
```

28. Be sure to set a password for Grub

```
┌──(ssb㉿kiss)-[~]
└─$ bash
ssb@kiss:~$
ssb@kiss:~$
ssb@kiss:~$
ssb@kiss:~$
ssb@kiss:~$ zsh
┌──(ssb㉿kiss)-[~]
└─$ 
```

> To protect my computer with a Linux distribution installed on it, it would be a good idea to set a password to access the GRUB boot loader.

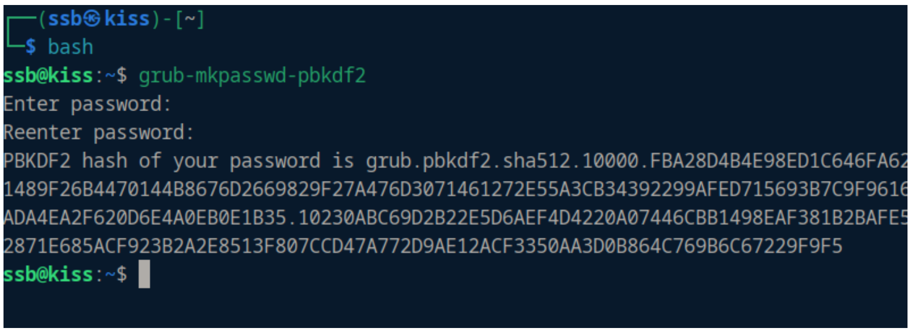
This is done in three steps:

1) Generate a password hash

I perform this action using the grub-mkpasswd-pbkdf2 utility . In Debian it is installed on the system by default. I run the command in the terminal:

```
~$ grub-mkpasswd-pbkdf2
```

I get:



```
┌──(ssb㉿kiss)-[~]
└─$ bash
ssb@kiss:~$ grub-mkpasswd-pbkdf2
Enter password:
Reenter password:
PBKDF2 hash of your password is grub.pbkdf2.sha512.10000.FBA28D4B4E98ED1C646FA62
1489F26B4470144B8676D2669829F27A476D3071461272E55A3CB34392299AFED715693B7C9F9616
ADA4EA2F620D6E4A0EB0E1B35.10230ABC69D2B22E5D6AEF4D4220A07446CBB1498EAF381B2BAFE5
2871E685ACF923B2A2E8513F807CCD47A772D9AE12ACF3350AA3D0B864C769B6C67229F9F5
ssb@kiss:~$ 
```

The resulting **hash** is a long string that starts with **grub.pbkdf2......**

I copy it into a text editor and save it into a separate document file on disk.

- Then I configure the **00_header** file located in **/etc/grub.d/** . I open it in a text editor and edit its contents as follows.

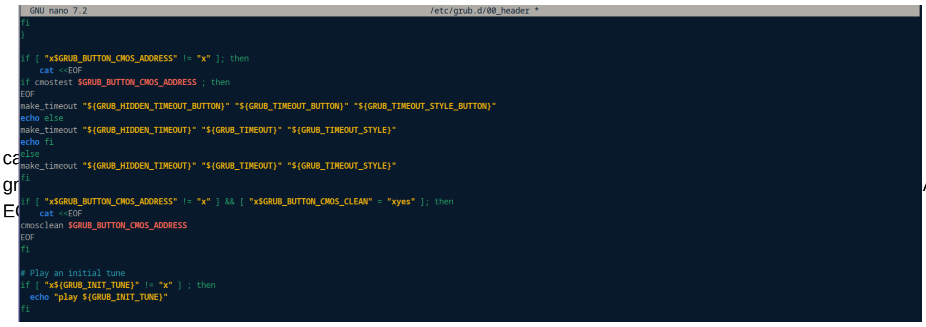
2) I assign a super user for GRUB

```
~$ sudo nano /etc/grub.d/00_header
```

- First, I open the document file in a text editor where I saved my **hash** , and insert my **hash** adding the missing lines so that the script looks like this:

- And I insert my **script prepared in advance at the very end of the file 00_header** opened in a text editor like this:

cat << EOF set superusers="мой_name" password_pbkdf2 мой_name grub.pbkdf2.sha512.10000.80FB9FC7543C7DE66BDE1255A50F26D4A0666A335E4758E84E1FE4EC183A75A52672F6D38383244DEF7784A3ACA30E2B6326ECCB557 EOF



```
GNU nano 7.2                    /etc/grub.d/00_header *
fi
}

if [ "x$GRUB_BUTTON_CMOS_ADDRESS" != "x" ]; then
    cat <<EOF
if cmostest $GRUB_BUTTON_CMOS_ADDRESS ; then
EOF
make_timeout "${GRUB_HIDDEN_TIMEOUT_BUTTON}" "${GRUB_TIMEOUT_BUTTON}" "${GRUB_TIMEOUT_STYLE_BUTTON}"
echo else
make_timeout "${GRUB_HIDDEN_TIMEOUT}" "${GRUB_TIMEOUT}" "${GRUB_TIMEOUT_STYLE}"
echo fi
else
make_timeout "${GRUB_HIDDEN_TIMEOUT}" "${GRUB_TIMEOUT}" "${GRUB_TIMEOUT_STYLE}"
fi

if [ "x$GRUB_BUTTON_CMOS_ADDRESS" != "x" ] && [ "x$GRUB_BUTTON_CMOS_CLEAN" = "xyes" ]; then
    cat <<EOF
cmosclean $GRUB_BUTTON_CMOS_ADDRESS
EOF
fi

# Play an initial tune
if [ "x${GRUB_INIT_TUNE}" != "x" ] ; then
  echo "play ${GRUB_INIT_TUNE}"
fi
```

ca
gr A52672F6D38383244DEF7784A3ACA30E2B6326ECCB557
EO

I copy it into a text editor and save it into a separate document file on disk.

- Then I configure the **00_header** file located in ***/etc/grub.d/*** . I open it in a text editor and edit its contents as follows.

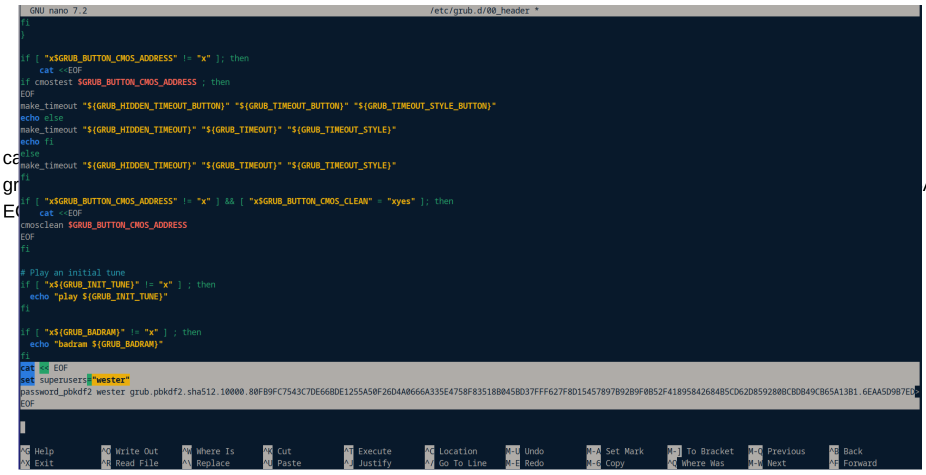
2) I assign a super user for GRUB

```
~$ sudo nano /etc/grub.d/00_header
```

- First, I open the document file in a text editor where I saved my **hash** , and insert my **hash** adding the missing lines so that the script looks like this:

- And I insert my **script prepared in advance at the very end of the file 00_header** opened in a text editor like this:

cat << EOF set superusers="мой_name" password_pbkdf2 мой_name grub.pbkdf2.sha512.10000.80FB9FC7543C7DE66BDE1255A50F26D4A0666A335E4758E84E1FE4EC183A75A52672F6D38383244DEF7784A3ACA30E2B6326ECCB557 EOF



```
  GNU nano 7.2                                     /etc/grub.d/00_header *
fi
}

if [ "x$GRUB_BUTTON_CMOS_ADDRESS" != "x" ]; then
    cat <<EOF
if cmostest $GRUB_BUTTON_CMOS_ADDRESS ; then
EOF
make_timeout "${GRUB_HIDDEN_TIMEOUT_BUTTON}" "${GRUB_TIMEOUT_BUTTON}" "${GRUB_TIMEOUT_STYLE_BUTTON}"
echo else
make_timeout "${GRUB_HIDDEN_TIMEOUT}" "${GRUB_TIMEOUT}" "${GRUB_TIMEOUT_STYLE}"
echo fi
else
make_timeout "${GRUB_HIDDEN_TIMEOUT}" "${GRUB_TIMEOUT}" "${GRUB_TIMEOUT_STYLE}"
fi

if [ "x$GRUB_BUTTON_CMOS_ADDRESS" != "x" ] && [ "x$GRUB_BUTTON_CMOS_CLEAN" = "xyes" ]; then
    cat <<EOF
cmosclean $GRUB_BUTTON_CMOS_ADDRESS
EOF
fi

# Play an initial tune
if [ "x${GRUB_INIT_TUNE}" != "x" ] ; then
  echo "play ${GRUB_INIT_TUNE}"
fi

if [ "x${GRUB_BADRAM}" != "x" ] ; then
  echo "badram ${GRUB_BADRAM}"
fi
cat << EOF
set superusers="wester"
password_pbkdf2 wester grub.pbkdf2.sha512.10000.80FB9FC7543C7DE66BDE1255A50F26D4A0666A335E4758F83518B045BD37FFF627F8D15457897B92B9F0B52F41895842684B5CD62D859280BCBDB49CB65A13B1.6EAA5D9B7ED>
EOF


^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark  M-] To Bracket M-Q Previous ^B Back
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy      ^Q Where Was   M-W Next     ^F Forward
```

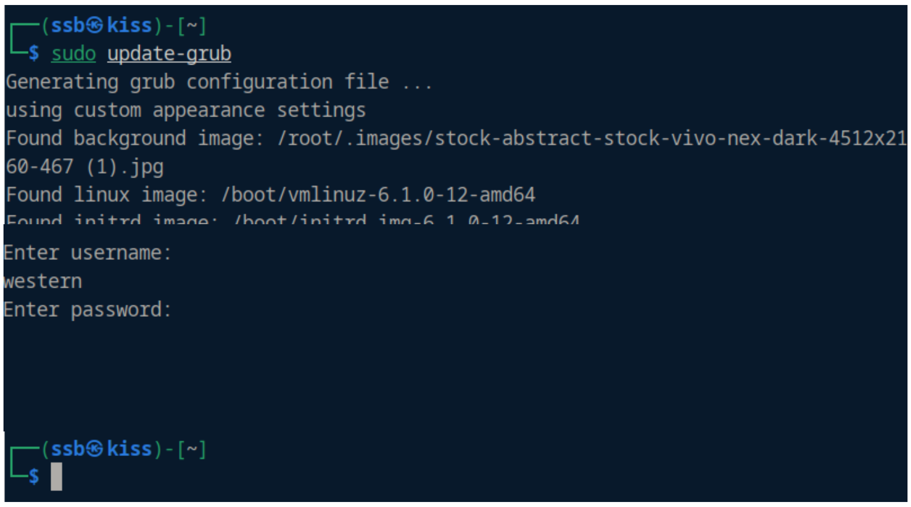
ca
gr A52672F6D38383244DEF7784A3ACA30E2B6326ECCB557
EO

3) I update the GRUB configuration

I do this to apply previously made changes using the command in the terminal:

```
~$ sudo update-grub
```

I get:



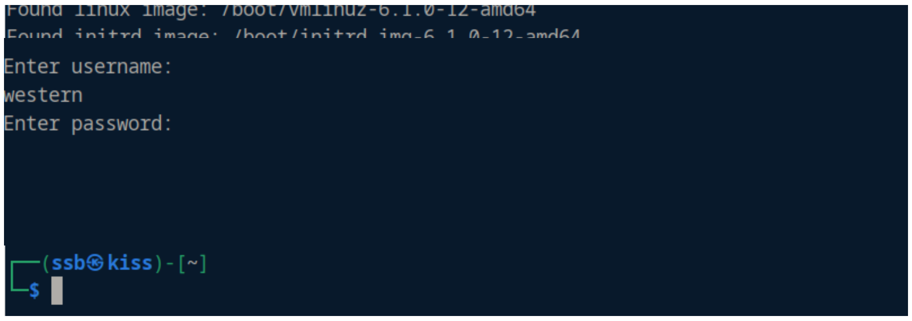
```
┌──(ssb㉿kiss)-[~]
└─$ sudo update-grub
Generating grub configuration file ...
using custom appearance settings
Found background image: /root/.images/stock-abstract-stock-vivo-nex-dark-4512x21
60-467 (1).jpg
Found linux image: /boot/vmlinuz-6.1.0-12-amd64
Found initrd image: /boot/initrd.img-6.1.0-12-amd64
Enter username:
western
Enter password:



┌──(ssb㉿kiss)-[~]
└─$ 
```

I reboot and get the welcome screen:

Final Word

My Debian 12 "Bookworm" configuration with the **Xfce** workspace gives me a great combination of performance and functionality, but it is my personal configuration, driven by my habits, my needs and my likes.

Each of you can have your own Debian 12 "Bookworm" and your own configuration.



```
Found linux image: /boot/vmlinuz-6.1.0-12-amd64
Found initrd image: /boot/initrd.img-6.1.0-12-amd64
Enter username:
western
Enter password:


┌──(ssb㉿kiss)-[~]
└─$ 
```

I reboot and get the welcome screen:

Final Word

My Debian 12 "Bookworm" configuration with the **Xfce** workspace gives me a great combination of performance and functionality, but it is my personal configuration, driven by my habits, my needs and my likes.

Each of you can have your own Debian 12 "Bookworm" and your own configuration.

Give **Debian 12 "Bookworm"** with **Xfce** a chance and you won't regret your choice! I will be glad to accept your comments and suggestions!

Authors

Sergei Sobolewski (https://www.linkedin.com/in/siergej-s-25a16319a/)

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2023-09-15	0.2	Siergej	Update Lab to Git, Bash
2023-09-08	0.1	Siergej	Created JupyterLab

© 2023. All rights reserved.