

# Linux CLI tips

- [Wifi connection](#)
- [Grub2 complete explanation](#)
- [My grub2 config](#)
- [Swap Ctrl-Left and Alt-Left on your keyboard](#)
- [BTRFS snapshots with snapper](#)
- [Create Services in Alpine Linux](#)
- [Almalinux LXC setup](#)

# Wifi connection

## Configure WiFi Connections

This article comes from [here](#).

This section explains how to establish a WiFi connection. It covers creating and modifying connections as well as directly connecting.

## Establish a Wireless Connection

This section will show how to establish a wifi connection to the wireless network. Note that directly connecting will implicitly create a connection (that can be seen with “nmcli c”). The naming of such will follow “SSID N” pattern, where N is a number.

First, determine the name of the WiFi interface:

```
$ nmcli d
```


DEVICE	TYPE	STATE	CONNECTION
...			
wlan0	wifi	disconnected	--

Make sure the WiFi radio is on (which is its default state):

```
$ nmcli r wifi on
```

Then, list the available WiFi networks:

```
$ nmcli d wifi list
```

* SSID	MODE	CHAN	RATE	SIGNAL	BARS	SECURITY
...						
my_wifi	Infra	5	54 Mbit/s	89		WPA2

As an example, to connect to the access point ‘my\_wifi’, you would use the following command:

```
$ nmcli d wifi connect my_wifi password <password>
```

is the password for the connection which needs to have 8-63 characters or 64 hexadecimal characters to specify a full 256-bit key.

# Connect to a Hidden Network

A hidden network is a normal wireless network that simply does not broadcast its SSID unless solicited. This means that its name cannot be searched and must be known from some other source.

Issue the following command to create a connection associated with a hidden network :

```
$ nmcli c add type wifi con-name <name> ifname wlan0 ssid <ssid>  
$ nmcli c modify <name> wifi-sec.key-mgmt wpa-psk wifi-sec.psk <password>
```

Now you can establish a connection by typing:

```
$ nmcli c up <name>
```

is an arbitrary name given to the connection and is the password to the network. It needs to have between 8-63 characters or 64 hexadecimal characters in order to specify a full 256-bit key.

## Further Information

You will find further information and more detailed examples on following page:

- <https://developer.gnome.org/NetworkManager/unstable/nmcli.html>

# Grub2 complete explanation

Tutoriel trouvé [sur cette page](#)

Table des matières

- **A savoir avant de configurer**
  - Fichier grub.cfg
  - Fichier /etc/default/grub
  - Dossier /etc/grub.d/
- **Paramétrage manuel**
  - Afficher/Masquer le menu
  - Intitulé dans le menu
  - Menu par défaut
  - Désactiver Memtest
  - os-prober partiel
  - Changer l'ordre des menus
  - Clavier français
  - Booter en automatique comme le choix précédent.
  - Ne pas installer la structure de boot.
- **Contenu de menu perso**
  - Chaîner un autre menu
  - Aérer le menu
  - Ancien paramètre vga=xxx
  - Menus pour lancer des images iso
  - Menu Arrêt du système
  - mise à jour
- **Habillage (mode texte)**
  - Résolution d'affichage
  - Fond d'écran
  - Couleurs d'affichage
- **THEME**
  - Sources

- [Où sont-ils installés ?](#)
- [Activation](#)
- [Autres pages en rapport](#)

[tutoriel](#), [grub-pc](#), [amorçage](#)

---

Cette page est un tutoriel Grub2 dont la documentation principale est disponible [ici](#).

# Tutoriel GRUB 2 : paramétrage manuel

Il existe des [applications graphiques pour paramétrer Grub](#). Ici on exposera l'**intervention directe** sur les **fichiers de paramètres** de Grub.

## A savoir avant de configurer

Nous n'intervenons **pas directement** sur un fichier de configuration **mais** sur des fichiers de **paramètres** qui sont pris en compte uniquement lors du lancement d'une **commande de mise à jour** (update-grub).

Les paramétrages sont **situés uniquement** dans le fichier **/etc/default/grub** et dans le dossier **/etc/grub.d/**.

## Fichier grub.cfg

- **/boot/grub/grub.cfg** : est un fichier de configuration généré automatiquement par *update-grub* (il est inutile d'éditer ce fichier) et **ne doit** donc **pas être modifié manuellement**.

## Fichier /etc/default/grub

Dans ce fichier, on peut **activer ou désactiver un paramètre** en le commentant / décommentant avec le caractère **dièse « # »**([croisillon](#)).

## paramètres présents par défaut

- **GRUB\_DEFAULT=0** correspond au menu qui sera sélectionné par défaut. (→ [Détails](#))
- **#GRUB\_HIDDEN\_TIMEOUT=0** : avec le « # », le menu de grub sera visible. Sans le « # », le menu de grub sera invisible. (→ [Détails](#))
- **GRUB\_HIDDEN\_TIMEOUT\_QUIET=false** De pair avec 'GRUB\_HIDDEN\_TIMEOUT'. (→ [Détails](#))
- **GRUB\_TIMEOUT\_STYLE=hidden** Paramètre apparu avec la version 18.04.1 en remplacement des deux qui précèdent.
- **GRUB\_TIMEOUT=10** est la durée en secondes de l'affichage du menu avant de se lancer sur le menu sélectionné par défaut. (→ [Détails](#))
- **GRUB\_DISTRIBUTOR=`** est la ligne qui définit la syntaxe des titres du menu (→ [Détails](#))
- **GRUB\_CMDLINE\_LINUX\_DEFAULT=" "** Des paramètres sont déjà présents, ils peuvent être ôtés. On y met les paramètres à ajouter lors de la détection automatique des systèmes lors du démarrage. (ex: radeon.modeset=1 logo.nologo ..).
- **GRUB\_CMDLINE\_LINUX=""** On peut y mettre des paramètres supplémentaires à ceux du paramètre précédant à ajouter lors du démarrage des systèmes en mode recovery (ex: fsck.mode=force fsck.repair=yes ...).
- **#GRUB\_GFXMODE=640x480** (→ [Détails](#)).
- **#GRUB\_INIT\_TUNE="480 440 1"** : sans le « # », on aura un bip à l'affichage du menu Grub.

## paramètres qu'on peut ajouter

- **GRUB\_GFXPAYLOAD=1024x768** : définit la résolution d'affichage entre Grub et celle définie dans la distribution (pour le splash screen par exemple). Défini à 'keep' par défaut.
- **GRUB\_GFXPAYLOAD\_LINUX=auto** : définit la résolution d'affichage par défaut du linux lancé image not found or type unknown
- **GRUB\_BACKGROUND="/boot/grub/images/fjord.jpg"** : pour mettre une image en fond d'écran (→ [Détails](#))
- **GRUB\_DISABLE\_OS\_PROBER="true"** : si on souhaite désactiver la recherche d'autres systèmes à chaque update-grub (inutile si on les lance autrement)
- **GRUB\_DISABLE\_OS\_PROBER=false** : si on souhaite activer la recherche d'autres systèmes à chaque update-grub pour les lancer.[avec le grub](#)
- **GRUB\_OS\_PROBER\_SKIP\_LIST="..."** : liste de partitions à ne pas inclure dans la recherche automatique de systèmes. (→ [Détails](#))
- **GRUB\_DISABLE\_LINUX\_RECOVERY="true"** : on mettra ce paramètre si on ne veut plus avoir la possibilité de lancer les systèmes en mode 'maintenance'
- **GRUB\_DISABLE\_SUBMENU=y** : si on souhaite voir directement une entrée de menu par noyau linux disponible (On peut supprimer les [noyaux](#) les plus anciens → [Nettoyer Ubuntu](#))

- **LANG=fr\_FR** : pour définir la langue utilisée pour l'édition du menu depuis Grub ou le mode 'ligne de commande'
- **GRUB\_TERMINAL\_INPUT=at\_keyboard** : nécessaire pour paramétrer le clavier français (pour ne plus avoir à chercher les (])[= .. du clavier Qwerty). (→ [Détails](#))
- **GRUB\_THEME="/boot/grub/themes/ubuntu-mate/theme.txt"** : chemin vers le fichier de configuration du thème (mode graphique). (→ [Détails](#))

## Dossier /etc/grub.d/

Ce dossier contient tous les scripts qui seront utilisés (en respectant l'ordre de numérotation) par update-grub pour créer le fichier grub.cfg.

- **00\_header** : script gérant les paramètres définis dans /etc/default/grub ;
- **05\_debian\_theme** : script pour gérer le thème en mode texte (fonds d'écran et couleurs) ;
- **10\_linux** : contient le script de lancement du système sur lequel on est ;
- **20\_memtest86+** : script permettant de générer les entrées memtest ; semble absent de certaines installations EFI
- **20\_linux\_xen** : script pour Xen Linux et Xen Hypervisor ;
- **30\_os-prober** : contient le script de recherche des autres systèmes installés ;
- **30\_uefi-firmware** : script pour localiser les paramètres uefi de l'ordinateur ;
- **40\_custom** : configuration personnelle (systèmes à lancer en configuration manuelle, paramètres qui n'existent pas pour /etc/default/grub,..)
- **41\_custom** : identique à 40\_custom (si on le renomme par exemple 07\_custom, on pourra y mettre le lancement du mediacenter ou de Windows en premier sur la liste)

**Nous n'utiliserons** pour nos paramétrages **que les fichiers 'custom'**.

## Paramétrage manuel

À partir de ce chapitre, vous allez intervenir directement sur les fichiers de GRUB. Soyez prudent, les fichiers de GRUB ne doivent pas être modifiés à la légère, sous peine d'avoir un système qui ne démarre plus. Si vous ne maîtrisez pas bien, n'hésitez pas à demander de l'aide sur le [forum](#). Pour revenir en arrière, utilisez un live-usb et suivez la procédure décrite dans cette [page de la doc](#).

Avant de commencer, il est souhaitable :

- de lancer dans un [terminal](#) une [commande](#) de mise à jour : `sudo update-grub` ;
- de créer une [sauvegarde](#) (ex. : `sudo cp /boot/grub/grub.cfg /boot/grub/grub.cfg.autogénéré`) ;

De même, aucun script de configuration présent dans **/etc/grub.d/** ne doit être effacé.  
*Par exemple, pour ne pas détecter les autres OS, il est préférable d'utiliser le paramètre `GRUB_DISABLE_OS_PROBER="true"` de `/etc/default/grub`.*

Pour tous nos paramétrages, on **éditera**, avec les **droits d'administration**, quasi **exclusivement les fichiers `/etc/default/grub` et `/etc/grub.d/40_custom`**.

**Une fois toutes les modifications opérées**, on lancera la **commande de mise à jour** :

```
sudo update-grub
```

## Afficher/Masquer le menu

Pour afficher ou non le menu Grub, on utilisera les paramètres **`GRUB_HIDDEN_TIMEOUT`** et **`GRUB_HIDDEN_TIMEOUT_QUIET`** du fichier **`/etc/default/grub`**.

### cas 1

#### Caché sans attendre le choix

Remarque : avec `GRUB_HIDDEN_TIMEOUT_QUIET=true`, impossible de mettre en pause le démarrage en tapotant la touche Shift lors de l'affichage du menu.

Pour Ubuntu 16.04 :

```
GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=true
GRUB_TIMEOUT=0 # dépendance: lorsque GRUB_HIDDEN_TIMEOUT est actif, ce paramètre doit être défini à 0
```

À partir d'Ubuntu 18.04 :

```
GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=0
```

### cas 2

**Caché avec un décompte de 5 secondes** pendant lequel on peut appuyer sur les touches **Échap** ou majuscule (**Shift**) pour afficher le menu. Sans action de notre part, le choix par défaut est lancé.

Remarque : puisque `GRUB_HIDDEN_TIMEOUT_QUIET=false`, alors on peut mettre en pause le démarrage en tapotant la touche Shift pour afficher le menu.

Pour Ubuntu 16.04 :



```
GRUB_HIDDEN_TIMEOUT=5
GRUB_HIDDEN_TIMEOUT_QUIET=false
GRUB_TIMEOUT=0# dépendance: lorsque GRUB_HIDDEN_TIMEOUT est actif, ce paramètre doit être défini à 0
```

À partir d'Ubuntu 18.04 :

```
GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=5
```

## cas 3

**Affiché directement avec un décompte de 10s.**

Pour Ubuntu 16.04 :

```
#GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=true
GRUB_TIMEOUT=10# ne pas laisser à zéro si vous voulez avoir le temps de sélectionner éventuellement un autre
```

À partir d'Ubuntu 18.04 :

```
GRUB_TIMEOUT_STYLE=menu
GRUB_TIMEOUT=10
```

## Intitulé dans le menu

**Par défaut**, la ligne du menu grub concernant la distribution sur laquelle vous êtes affiche « **Ubuntu GNU/Linux** ». Que vous soyez sur une **variante** (Xubuntu, Lubuntu, Kubuntu,..) ou sur un **dérivé** (Linux Mint,..), l'intitulé reste **le même** car il dépend de la commande 'lsb\_release'.

Voici comment on peut influencer ce comportement en modifiant le paramètre **GRUB\_DISTRIBUTOR** du fichier **/etc/default/grub** selon les possibilités suivantes :

```
GRUB_DISTRIBUTOR=`lsb_release -is 2> /dev/null || echo Debian`# donne : Ubuntu GNU/Linux
GRUB_DISTRIBUTOR=`lsb_release -ds 2> /dev/null || echo Debian`# donne : Ubuntu 16.04.1 LTS GNU/Linux
GRUB_DISTRIBUTOR=`echo -n $(lsb_release -cds 2> /dev/null || echo Debian)`# donne : Ubuntu 16.04.1 LTS xe
GRUB_DISTRIBUTOR=`echo -n TITRE PERSONNALISÉ $(lsb_release -rs 2> /dev/null || echo Debian)`# donne : TITR
GRUB_DISTRIBUTOR=`echo -n TITRE PERSONNALISÉ`# donne : TITRE PERSONNALISÉ GNU/Linux
```

## Menu par défaut

On utilisera ici les paramètres **GRUB\_DEFAULT** et **GRUB\_TIMEOUT** du fichier **/etc/default/grub**

Pour changer le système d'exploitation sur lequel grub démarre par défaut on paramètrera **GRUB\_DEFAULT=** au choix avec :

- un **chiffre** donnant la position de la ligne à sélectionner dans la liste.(0 = le premier, 1 = le deuxième, etc.) ;
- **saved** (sans guillemet) pour ce qui a été sélectionné lors du précédent lancement (nécessite de rajouter GRUB\_SAVEDEFAULT=true) ;
- ou le **titre** de menu exact d'un menu mis **entre guillemets** (par exemple : "Microsoft Windows 10 (on /dev/sda1)"). Pour obtenir les titres exacts des menus : `grep menuentry /boot/grub/grub.cfg` ;
- ou le titre du menu avancé et le titre exact du sous-menu séparés par le caractère **>**. Le tout mis entre guillemets. Par exemple: `GRUB_DEFAULT="Options avancées pour Ubuntu>Ubuntu, avec Linux 4.15.0-176-generic"`
- "**chiffre>chiffre**" le premier chiffre décrit un menu principal, le second chiffre décrit le sous-menu (Numérotation à partir de 0) . Ils doivent aussi être mis **entre guillemets**. Par exemple "1>2" pour le mode normal de l'ancien noyau.

La **temporisation** avant un lancement automatique se définit avec le paramètre **GRUB\_TIMEOUT** :

- **GRUB\_TIMEOUT=10**, lancera le système par défaut au bout de **10 secondes**. Si on ne veut **pas de temporisation**, on mettra : **GRUB\_TIMEOUT=-1**

## Désactiver Memtest

Si on ne souhaite plus voir dans les menus Memtest, on **changera le statut du script** pour qu'il **ne soit plus exécutable** à la prochaine commande de mise à jour :

```
sudo chmod -x /etc/grub.d/20_memtest86+
```

Pour le réactiver, on mettra '+x' au lieu de '-x' avec la même commande.

## os-prober partiel

On peut demander à grub d'**éviter de scanner certaines partitions** par leur UUID (`sudo blkid` pour avoir leur liste). On utilisera le paramètre **GRUB\_OS\_PROBER\_SKIP\_LIST** du fichier **/etc/default/grub** :

```
GRUB_OS_PROBER_SKIP_LIST="3a43c682-cb67-47e6-83cf-e647a72abb5d@/dev/sda3","1397cb72-27f1-4e01-acaf-8b123456789a@/dev/sdb1"
```

**ATTENTION** : Il semble que la bonne syntaxe soit :

```
GRUB_OS_PROBER_SKIP_LIST="3a43c682-cb67-47e6-83cf-e647a72abb5d@/dev/sda3 1397cb72-27f1-4e01-acaf-8b123456789a@/dev/sdb1"
```

# Changer l'ordre des menus

On peut **renommer les scripts** utilisés par 'update-grub' dans le dossier /etc/grub.d/ pour en changer l'ordre, par exemple :

- **sudo mv /etc/grub.d/30\_os-prober /etc/grub.d/06\_os-prober** (par exemple pour mettre Windows en premier sur le menu)

On peut **copier les entrées des menus dans 40\_custom** (depuis /boot/grub/grub.cfg) dans l'ordre qu'on souhaite **et désactiver la détection automatique** en mettant le paramètre GRUB\_DISABLE\_OS\_PROBER="true" dans /etc/default/grub. On peut renommer 41\_custom en 07\_custom pour qu'il se place avant 10\_linux (on y mettra le menu qu'on veut en premier).

## Clavier français

Quand on édite un menu, c'est galère de retrouver la ponctuation, les parenthèses ou encore le signe égal sur un clavier qwerty. La solution est de paramétrer la disposition azerty :

- testé avec le **clavier 'Français (variante)' actif**. On commencera par **créer** le dossier **/boot/grub/layouts**. Ensuite, on **générera** la disposition du clavier dans un **fichier reconnu par grub** :

```
sudo mkdir /boot/grub/layouts
sudo grub-kbdcomp -o /boot/grub/layouts/fr.gkb fr
```

- On **ajoutera** le paramètre **GRUB\_TERMINAL\_INPUT=at\_keyboard** au fichier **/etc/default/grub**
- On **ajoutera** ces lignes au fichier **/etc/grub.d/40\_custom** :

```
# Clavier fr
insmod keylayouts
keymap fr
```

## Booter en automatique comme le choix précédent.

```
GRUB_DEFAULT=saved
GRUB_SAVEDEFAULT=true
```

## Ne pas installer la structure de boot.

Ceci est un ajout du 15/08/2017 dont le contenu demande encore à être confirmé. Lorsqu'on dispose de plusieurs OS ubuntu et qu'on passe souvent de l'un à l'autre et que les noyaux se mettent à jour, on arrive rapidement à avoir un énorme fichier boot.cfg. Afin de l'épurer, il existe un [outil](#)

L'idéal étant de ne pas arriver à cette situation. Il existe quelques palliatifs pour retarder cet état de fait. Pour tous les OS qui ne sont pas l'OS dirigeant (souvent la version LTS):

IL faut demander à ne faire aucune recherche d'OS complémentaires via la commande `<code>sudo chmod -x /etc/grub.d/30_os-prober</code>` il est preferable de modifier le fichier `"/etc/default/grub "` et d' y inserer l ' option apropiée :

```
GRUB_DISABLE_OS_PROBER=true
```

Il faut aussi ne pas ré-écraser le démarrage du boot prioritaire. Ce qui se fait à chaque fois qu'un nouveau noyau arrive. Je n'ai pas vu d'option dans ce fichier de paramétrage pour le faire. En attendant:

```
sudo chmod -x /usr/sbin/grub-install
```

# Contenu de menu perso

Tous ces contenus sont ajoutés à `/etc/grub.d/40_custom`

## Chaîner un autre menu

**chainloader** vous permettra de lancer un autre gestionnaire d'amorçage présent sur le premier secteur des partitions concernées : **Windows, Lilo...** Ça ne fonctionne pas toujours bien donc on préfère souvent une autre alternative. Voici quelques exemples.

Remarque : l'entête du fichier `40_custom` doit comporter ces lignes :

```
#!/bin/sh
exec tail -n +3 $0
```

```
menuentry "Windows 10" {
  insmod ntfs
  search --set=root --label WINDOWS_10 --hint hd0,msdos2
  ntldr /bootmgr
}
menuentry "Windows 10b" {
  insmod ntfs
```

```

search --set=root --label WINDOWS_10b --hint hd0,msdos2
ntldr /bootmgr
}
menuentry "Windows 7" {
    insmod ntfs
    set root='(hd0,3)'
    search --no-floppy --fs-uuid --set 94E84428E8440B46
    chainloader +1
}
menuentry "Windows 10 en mode bios legacy" {
    insmod ntfs
    set root='(hd1,msdos2)'
    chainloader +1
}
menuentry 'Bootloader Lilo sur sda7' {
    set root=(hd0,7)
    chainloader +1
}
menuentry "Menu grub2 sur sda8" {
    set root=(hd0,8)
    configfile /boot/grub/grub.cfg
}
menuentry "Menu grub-legacy sur sda6" {
    set root=(hd0,6)
    legacy_configfile /boot/grub/menu.lst
}

```

## Aérer le menu

Pour ajouter des **lignes vides**, de **sous-titres**,... on procédera ainsi avec juste un **'true'** pour qu'elles soient **prises en compte** :

```

## ligne vide
menuentry " " {
    true
}
## ligne de sous-titre
menuentry "----- Dérivés Ubuntu -----" {
    true
}

```

## Ancien paramètre vga=xxx

Dans les options de boot d'un Linux, on trouve parfois le **paramètre « vga=788 »** avec une définition d'écran donné. Consultez [ce tableau](#) pour pouvoir ajouter un 'set gfxpayload' équivalent dans votre 'menuentry'.

Donc, par exemple, au lieu de **vga=788** dans la ligne 'linux', on ajoutera au-dessus une ligne **set gfxpayload=800x600** à la place.

## Menus pour lancer des images iso

Dans ce tutoriel vous aurez des exemples de menus « prêts à l'emploi » pour lancer les livecd Ubuntu ou autres depuis leur fichier iso :

- **Lancer des images iso directement depuis GRUB 2**

## Menu Arrêt du système

```
menuentry 'Arrêt du système' {
    halt
}
menuentry 'Redémarrage du système' {
    reboot
}
```

## mise à jour

Comme toujours, après tous ces paramétrages, on lancera la commande de mise à jour :

```
sudo update-grub
```

## Habillage (mode texte)

## Résolution d'affichage

### vbeinfo

Quand on est sur le menu grub, on **appuie sur « c »**. On est ainsi en **ligne de commande**. On lancera alors la commande :

```
vbeinfo
```

La commande ***vbeinfo*** est remplacée par ***videoinfo***

Toutes les **résolutions supportées par grub** s'affichent. La plus haute est la dernière sur la liste.  
On la note (par exemple: 1280x1024x32)

## GRUB\_GFXMODE

Sur le fichier **/etc/default/grub**, on modifiera le paramètre GRUB\_GFXMODE qui définit la **résolution d'affichage pour Grub** ainsi (sans « # » devant) :

```
GRUB_GFXMODE=1280x1024,1024x768x32
```

La deuxième valeur sera prise en compte si la première n'est pas supportée par votre matériel. Le x32 est la profondeur de codage des couleurs en bits.

Si le démarrage se bloque sur un écran noir ou sur l'image de fond, appuyez sur 'c' pour passer en ligne de commande et saisissez, par exemple :

```
set gfxpayload=1024x768
```

## GRUB\_GFXPAYLOAD

On fera la même chose qu'au paragraphe précédent pour les résolutions après Grub avec les paramètres **GRUB\_GFXPAYLOAD** et **GRUB\_GFXPAYLOAD\_LINUX**.

# Fond d'écran

## Pré-requis

- **GRUB\_GFXMODE** doit être défini (→ [voir plus haut](#))
- une image dans un **format compatible** avec la capacité d'affichage de grub (→ [voir plus haut](#))

## Convertir l'image

Si vous avez **Gimp d'installé** ou au moins **ImageMagick**, vous pourrez lancer cette commande de conversion en l'adaptant à vos besoins :

```
convert ~/Images/"image exemple.jpg" -resize 1280x1024! -depth 16 ~/Bureau/"00_image_de_fond.jpg"  
sudo mv ~/Bureau/"00_image_de_fond.jpg" /boot/grub/.
```

## Utilisation

Pour l'utiliser, on ajoutera le paramètre **GRUB\_BACKGROUND** au fichier **/etc/default/grub**

```
GRUB_BACKGROUND="/boot/grub/00_image_de_fond.jpg"
```

Lancer ensuite la commande de mise à jour :

```
sudo update-grub
```

Si le fond d'écran n'est pas détecté lors de cette mise à jour, c'est qu'il n'est pas dans un format compatible pour Grub.

## Couleurs d'affichage

Pour définir les couleurs utilisées par grub, on ajoutera au fichier **/etc/grub.d/40\_custom** les lignes suivantes :

```
set color_normal=light-gray/black[]# définit les couleurs de texte/fond d'écran autour du cadre de menu
set menu_color_normal=light-gray/black[]# définit les couleurs de texte/fond d'écran dans le cadre de menu
set menu_color_highlight=yellow/light-blue[]# définit les couleurs de texte/surbrillance pour la ligne de menu sélectionnée
```

Les couleurs à notre disposition sont : black, blue, brown, cyan, dark-gray, green, light-cyan, light-blue, light-green, light-gray, light-magenta, light-red, magenta, red, white, yellow. 'black' devient une couleur transparente si on a une image de fond.

Lancer ensuite la commande de mise à jour :

```
sudo update-grub
```

## THEME

Ce sont des thèmes prêts à l'emploi. **Attention** tout de même à l'**image incluse**

**(background.png)** qui **doit**, **comme indiqué précédemment**, **être compatible** avec la résolution d'affichage de votre Grub.

On peut tester sa compatibilité en la définissant comme fond d'écran (→ [voir plus haut](#)) et en regardant si elle est détectée par la commande de mise à jour.

## Sources



On peut installer le theme inclus dans les dépôts : [grub2-themes-ubuntu-mate](#). On en trouve aussi [ici](#) (en cherchant bien 'themes Grub2' et non pas seulement 'gfx')

## Où sont-ils installés ?

Par défaut, ils sont installés dans le dossier **/boot/grub/themes**.

## Activation

Dans le fichier **/etc/default/grub**, on renseignera le paramètre **GRUB\_THEME** ainsi :

```
GRUB_THEME="/boot/grub/themes/ubuntu-mate/theme.txt"
```

Il s'agit du chemin vers le fichier de configuration. Il peut très bien porter un autre nom (par exemple theme-fr.txt sur un autre theme).

Lancer ensuite la commande de **mise à jour** :

```
sudo update-grub
```

## Autres pages en rapport

- [Doc principale de Grub2](#)
- [Grub Customizer : une interface graphique gérant Grub 2 et permettant de le personnaliser,](#)
- [Tutoriel pour lancer des images iso directement depuis GRUB 2.](#)
- [Récupérer Ubuntu après installation de Windows](#)

---

Contributeurs : [Frombenny](#) (rédaction de la page), [Theozzfancometh](#), [f.x0](#), [albanmartel](#), [perco](#), [floxyz51](#).

# My grub2 config

## Enable OS prober

If os-prober is disabled, add this line in `/etc/default/grub`:

```
GRUB_DISABLE_OS_PROBER=false
```

## Add `poweroff` and `reboot` options

To add `poweroff` and `reboot` options to your grub, add those lines in `/etc/grub.d/40_custom`:

```
menuentry 'Power off' {  
    halt  
}  
menuentry 'Reboot' {  
    reboot  
}
```

## Install a theme (link to the [github repo](#))

`banner` not found or type unknown

## Installation:

Usage: `sudo ./install.sh [OPTIONS...]`

-t, --theme	theme variant(s)	[tela vimix stylish whitesur]	(default is tela)
-i, --icon	icon variant(s)	[color white whitesur]	(default is color)
-s, --screen	screen display variant(s)	[1080p 2k 4k ultrawide ultrawide2k]	(default is 1080p)
-c, --custom-resolution	set custom resolution	(e.g., 1600x900)	(disabled in default)
-r, --remove	remove theme	[tela vimix stylish whitesur]	(must add theme name option, default is tela)
-b, --boot	install theme into '/boot/grub' or '/boot/grub2'		
-g, --generate	do not install but generate theme into chosen directory		(must add your directory)
-h, --help	show this help		

If no options are used, a user interface `dialog` will show up instead

## Examples:

- Install Tela theme on 2k display device:

```
sudo ./install.sh -t tela -s 2k
```

- Install Tela theme with custom resolution:

```
sudo ./install.sh -t tela -c 1600x900
```

- Install Tela theme into /boot/grub/themes:

```
sudo ./install.sh -b -t tela
```

- Uninstall Tela theme:

```
sudo ./install.sh -r -t tela
```

## Issues / tweaks:

### Correcting display resolution:

- On the grub screen, press `c` to enter the command line
- Enter `vbeinfo` or `videoinfo` to check available resolutions
- Open `/etc/default/grub`, and edit `GRUB_GFXMODE=[height]x[width]x32` to match your resolution
- Finally, run `grub-mkconfig -o /boot/grub/grub.cfg` to update your grub config

## Setting a custom background:

- Make sure you have `imagemagick` installed, or at least something that provides `convert`
- Find the resolution of your display, and make sure your background matches the resolution
  - 1920x1080 >> 1080p
  - 2560x1080 >> ultrawide
  - 2560x1440 >> 2k
  - 3440x1440 >> ultrawide2k
  - 3840x2160 >> 4k
- Place your custom background inside the root of the project, and name it `background.jpg`
- Run the installer like normal, but with `-s [YOUR_RESOLUTION]` and `-t [THEME]` and `-i [ICON]`
  - Make sure to replace `[YOUR_RESOLUTION]` with your resolution and `[THEME]` with the theme
- Alternatively, use the `-c` option to set a custom resolution

## Contributing:

- If you made changes to icons, or added a new one:
  - Delete the existing icon, if there is one
  - Run `cd assets; ./render-all.sh`
- Create a pull request from your branch or fork
- If any issues occur, report then to the [issue](#) page

## Preview:

preview  
page not found or type unknown

## Documents

[Grub2 theme reference](#)

[Grub2 theme tutorial](#)

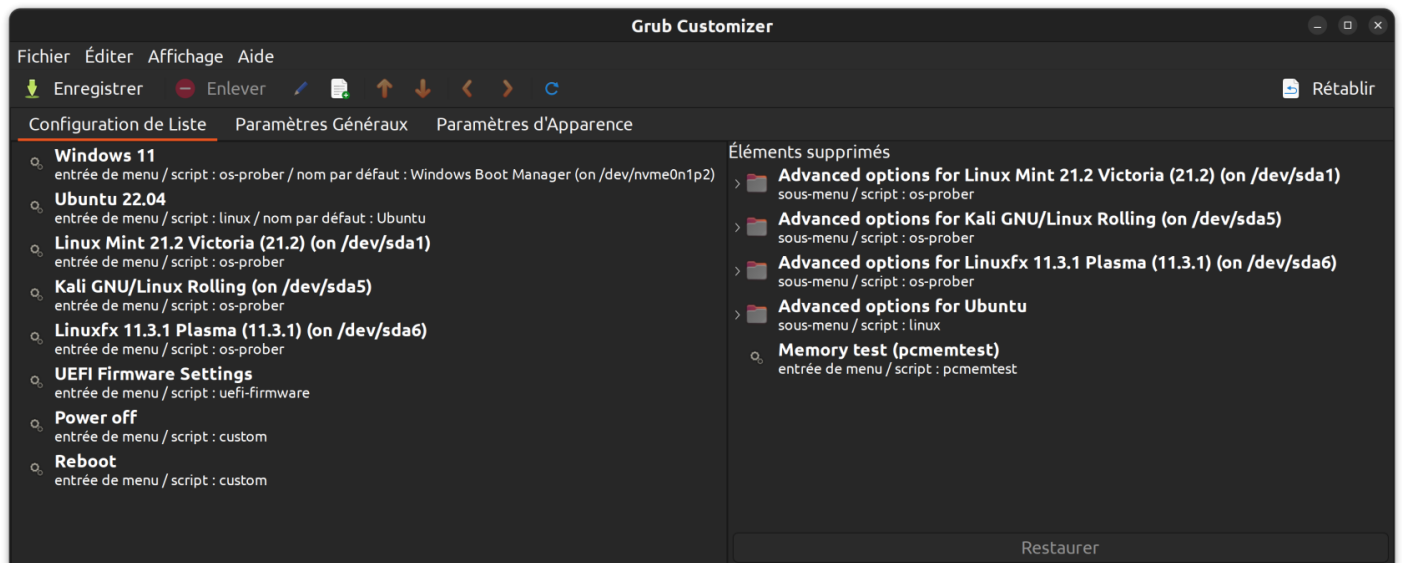
# Customize the titles, order and entries shown with grub-

# customizer

To install `grub-customizer`:

```
sudo add-apt-repository ppa:danielrichter2007/grub-customizer
sudo apt-get update
sudo apt-get install grub-customizer
```

Preview:



# Swap Ctrl-Left and Alt-Left on your keyboard

## Introduction

I know, not everyone is a fan of Apple, but I enjoyed quite a bit the UX of my old MacBook Pro, so I always swap the left `Ctrl` key with the left `Alt` key, like the `Command` key and `Option` key from Apple.

The thing I liked about that is the simplicity of using the `Ctrl` shortcuts with the thumb without moving the left hand that much... a personal preference, but still enjoyable IMO.

But swapping keys is not always an easy task, because, if it only works partially, it may completely break your interactions and shortcuts... definitely not a good experience.

## Configuration

The easiest and most compatible way I found is the following (it worked for me on Ubuntu and Archlinux):

1. Open a terminal
2. Type `nano ~/.Xmodmap` and then `Enter` (you can use whatever text editor you prefer: `vi`, `vim`, `nvim`, `gedit`...)
3. Type the following lines:

```
clear control
clear mod1
keycode 37 = Alt_L Meta_L
keycode 64 = Control_L
add control = Control_L Control_R
add mod1 = Alt_L Meta_L
```

4. Type `Ctrl-X`, then `Y` and then `Enter` (it may differ according to the text editor of your choice (`Escape`, `:wq` and `Enter` for `vi`, `vim` and `nvim`)).
5. Finally type `xmodmap ~/.Xmodmap` to apply the new configuration.

If it doesn't work for you, it may be because of a couple of reasons:

- your keyboard is not mapped the same way (different keycodes)
- you don't have `xmodmap`
- another reason I don't know of...

If the reason is the first one, you may be interested to see [that post](#) (where I found that tip).

# BTRFS snapshots with snapper

Picked from an [article here](#).

Set up automatic snapshots of a BTRFS root subvolume, add these snapshots to the GRUB boot menu, and gain the ability to rollback an [Arch Linux](#) system to an earlier state.

## Let's go!

See "[A\(rch\) to Z\(ram\)](#)" for my step-by-step install of Arch, where I created:

- `@` subvolume, mounted to `/`. Create snapshots of this root subvolume.
- `@snapshots` and other subvolumes, which are *excluded* from root snapshots.

## 1. Install Snapper and snap-pac

**Snapper** is a tool for managing BTRFS snapshots. It can create and restore snapshots, and provides scheduled auto-snapping. **Snap-pac** provides a Pacman hook that uses Snapper to create `pre-` and `post-` BTRFS snapshots triggered by use of the package manager.

Install ...

```
$ sudo pacman -S snapper snap-pac
```

## 2. Create snapshot configuration for root subvolume

Snapper's `create-config` command assumes:



- Subvolume `@` already exists and is mounted at `/`.
- `/.snapshots` directory is *not* mounted and doesn't exist.

During my Arch install, I created the `@` and `@snapshots` subvolumes, and `/.snapshots` mountpoint. Before letting Snapper do its config thing, I need to move my earlier snapshot setup out of the way.

Unmount the subvolume and remove the mountpoint ...

```
$ sudo umount /.snapshots
$ sudo rm -rf /.snapshots
```

Create a new `root` config ...

```
$ sudo snapper -c root create-config /
```

This generates:

- Configuration file at `/etc/snapper/configs/root`.
- Add `root` to `SNAPPER_CONFIGS` in `/etc/conf.d/snapper`.
- Subvolume `.snapshots` where future snapshots for this configuration will be stored.

## 3. Setup `/.snapshots`

List subvolumes ...

```
$ sudo btrfs subvolume list /
ID 256 gen 199 top level 5 path @
ID 257 gen 186 top level 5 path @home
ID 258 gen 9 top level 5 path @snapshots
[...]
ID 265 gen 199 top level 256 path .snapshots
```

Note the `@snapshots` subvolume I had created earlier, and the `.snapshots` created by Snapper.

I prefer my `@snapshots` setup over `.snapshots`, so I delete the Snapper-generated subvolume ...

```
$ sudo btrfs subvolume delete .snapshots
Delete subvolume (no-commit): '//.snapshots'
```

Re-create and re-mount `/.snapshots` mountpoint ...

```
$ sudo mkdir /.snapshots
```

```
$ sudo mount -a
```

This setup will make all snapshots created by Snapper be stored outside of the `@` subvolume. This allows replacing `@` without losing the snapshots.

Set permissions. Owner must be `root`, and I allow members of `wheel` to browse through snapshots ...

```
$ sudo chmod 750 /.snapshots
```

```
$ sudo chown :wheel /.snapshots
```

## 4. Manual snapshot

Example of taking a manual snapshot of a fresh install ...

```
$ sudo snapper -c root create -d "***Base system install**"
```

## 5. Automatic timeline snapshots

Setup timed auto-snapshots by modifying `/etc/snapper/configs/root`.

Allow user (example: `foo`) to work with snapshots ...

```
ALLOW_USERS="foo"
```

Example: Set some timed snapshot limits ...

```
TIMELINE_MIN_AGE="1800"
```

```
TIMELINE_LIMIT_HOURLY="5"
```

```
TIMELINE_LIMIT_DAILY="7"
```

```
TIMELINE_LIMIT_WEEKLY="0"
```

```
TIMELINE_LIMIT_MONTHLY="0"
```

```
TIMELINE_LIMIT_YEARLY="0"
```

Start and enable `snapper-timeline.timer` to launch the automatic snapshot timeline, and `snapper-cleanup.timer` to periodically clean up older snapshots...

```
$ sudo systemctl enable --now snapper-timeline.timer
```

```
$ sudo systemctl enable --now snapper-cleanup.timer
```

## 6. Pacman snapshots

Pacman `pre-` and `post-` snapshots are triggered before and after a significant change (such as a system update).

Example: I install `tree`, which triggers a `pre` and `post` install snapshot.

List configs ...

```
$ snapper list-configs
```

```
Config | Subvolume
```

```
-----+-----
```

```
root  | /
```

List snapshots taken for `root` ...

```
$ snapper -c root list
```

#	Type	Pre #	Date	User	Cleanup	Description	Userdata
0	single			root	current		
1	single		Sat 20 Aug 2022 11:21:53 AM	root		**Base system install**	
2	pre		Sat 20 Aug 2022 11:22:39 AM	root	number	pacman -S tree	
3	post	2	Sat 20 Aug 2022 11:22:39 AM	root	number	tree	
4	single		Sat 20 Aug 2022 12:00:04 PM	root	timeline	timeline	

List updated subvolumes list, which now includes the snapshots ...

```
$ sudo btrfs subvolume list /
```

```
ID 256 gen 270 top level 5 path @
```

```
ID 257 gen 270 top level 5 path @home
```

```
ID 258 gen 257 top level 5 path @snapshots
```

```
[...]
```

```
ID 266 gen 216 top level 258 path @snapshots/1/snapshot
```

```
ID 267 gen 218 top level 258 path @snapshots/2/snapshot
```

```
ID 268 gen 219 top level 258 path @snapshots/3/snapshot
```

```
ID 269 gen 237 top level 258 path @snapshots/4/snapshot
```

## 7. Updatedb

If `locate` is installed, skip indexing `.snapshots` directory by adding to `/etc/updatedb.conf` ...

```
PRUNENAMES = ".snapshots"
```

## 8. Grub-btrfs

Include the snapshots as boot options in the GRUB boot loader menu.

Install ...

```
$ sudo pacman -S grub-btrfs
```

Set the location of the directory containing the `grub.cfg` file in `/etc/default/grub-btrfs/config`.

Example: My `grub.cfg` is located in `/efi/grub` ...

```
GRUB_BTRFS_GRUB_DIRNAME="/efi/grub"
```

## 9. Auto-update GRUB

Enable `grub-btrfs.path` to auto-regenerate `grub-btrfs.cfg` whenever a modification appears in `/.snapshots` ...

```
$ sudo systemctl enable --now grub-btrfs.path
```

At the next boot, there is an submenu in GRUB for `Arch Linux snapshots`.

## 10. Read-only snapshots and overlays

Booting on a snapshot is done in *read-only* mode.

This can be tricky:

An elegant way is to boot this snapshot using **overlayfs** ... Using overlayfs, the booted snapshot will behave like a live-cd in non-persistent mode. The snapshot will not be modified, the system will be able to boot correctly, because a writeable folder will be included in the RAM ... Any changes in this system thus started will be lost when the system is rebooted/shutdown.

Add the hook `grub-btrfs-overlayfs` at the end of `HOOKS` in `/etc/mkinitcpio.conf` ...

```
HOOKS=(base ... fsck grub-btrfs-overlayfs)
```

Re-generate initramfs ...

```
$ sudo mkinitcpio -P
```

Note: Any snapshots that do not include this modified initramfs will not be able to use overlayfs.

## 11. System rollback the 'Arch Way'

Snapper includes a rollback tool, but on Arch systems the preferred method is a manual rollback.

After booting into a snapshot mounted `rw` courtesy of overlayfs, mount the toplevel subvolume (subvolid=5). That is, omit any subvolid or subvol mount flags (*example*: an encrypted device map labelled `cryptdev`) ...

```
$ sudo mount /dev/mapper/cryptdev /mnt
```

Move the broken `@` subvolume out of the way ...

```
$ sudo mv /mnt/@ /mnt/@.broken
```

Or simply delete the subvolume ...

```
$ sudo btrfs subvolume delete /mnt/@
```

Find the number of the snapshot that you want to recover ...

```
$ sudo grep -r '<date>' /mnt/@snapshots/*/info.xml
[...]  
/.snapshots/8/info.xml: <date>2022-08-20 15:21:53</date>  
/.snapshots/9/info.xml: <date>2022-08-20 15:22:39</date>
```

Create a read-write snapshot of the read-only snapshot taken by Snapper ...

```
$ sudo btrfs subvolume snapshot /mnt/@snapshots/number/snapshot /mnt/@
```

Where `number` is the snapshot you wish to restore as the new `@`.

Unmount `/mnt`.

Reboot and rollback!

# Create Services in Alpine Linux

I began my development adventure, as everyone else, with simple CLI apps (mostly incredibly useless...), and my next step was with web development.

First with `Golang`, then `Node.js` and then others.

Little by little, I learned to create decent-looking websites and I wanted to deploy them in my homelab. First, I did it on my first dedicated server, a `Raspberry Pi 5 8Gb` and it worked great! But then I bought a `Dell Poweredge Rack Server` to have a server hosting VMs.

There, I wanted to deploy my websites (my Portfolio and other little projects that could be of interest), but I wanted to keep the VMs really simple and lightweight, so I chose `Alpine Linux`, a great Linux distribution that keeps things quite essential. I knew of it because it's one of the most used images with Docker, so it explains its optimization.

But, what if the VM crashes or needs to be rebooted, or, simply, if I need to reboot the Server?

Then I would need to restart manually all web servers in the many VMs... a lot of hassle and possible issues.

The answer to that problematic is to **create a service that is started at boot time**.

In `systemd` distributions, you need to create a **service file**, but `Alpine Linux` uses `OpenRC` instead.

## Creating an `OpenRC` file

`OpenRC` files are located in `/etc/init.d/`, so first, you need to create a file there (the name of the file will be the name of the service):

```
sudo vim /etc/init.d/webserver
```

Then write your `OpenRC` file:

```
#!/sbin/openrc-run
```

```
# Service description
#
# This service starts the web server application
name=$RC_SVNAME
pidfile="/run/${RC_SVNAME}.pid"
# switch webserver to the name of your service in the two following lines (they are both optional configurations)
output_log="/var/log/webserver.log"
error_log="/var/log/webserver.err"

# Command needed to run the application
command="/path/to/your/webserver/executable"
# you can put all arguments below:
command_args="arg1 arg2 arg3"

# Command settings
command_background=true
# specify the user you wish to execute your webserver as:
command_user="www-data:www-data"

# Dependencies (most likely the network if it's a webserver, and there might be more if your application is more
demanding)
depend() {
    need net
}
```

Make the file executable, otherwise, you'll not be able to run anything and the start command will fail:

```
sudo chmod 755 /etc/init.d/webserver
```

### Warning!

You might need to give writing access to the user defined in the `command_user` line for the `output_log` and `error_log` files if you decide to specify them (they are optional)

## Using Environment variables



If you need to use environment variables in your `OpenRC`, you need to create a configuration file in `/etc/conf.d/` with the same name as the `OpenRC` file.

In it you can **export** all environment variables you might need, for example:

```
export WEBSERVER_PORT=4000
```

and then you can call it in the `OpenRC` file, for example:

```
command_args="port=${WEBSERVER_PORT}"
```

# Managing the **service** created

Consult that [Official documentation](#) from the Alpine Linux website if you wish to know more about it from the official page.

To manipulate your service, you can use those three commands (replace `webserver` with the name of the service):

```
rc-service webserver start
```

```
rc-service webserver status
```

```
rc-service webserver stop
```

To enable your service at the `default` level (when booting the system normally):

Replace `webserver` with the name of the service.

```
rc-update add webserver default
```

To remove your service from all levels:

```
rc-update delete webserver -a
```

With that, you should be able to create your services and execute them automatically at boot time in your `Alpine Linux` servers!

Happy coding!

# Almalinux LXC setup

Here is a simple review of what should be done in a newly created LXC using the image from Proxmox VE.

## 1. Install important packages

- update and upgrade

```
dnf update -y && dnf upgrade -y
```

- install basic packages

```
dnf install -y vim git wget net-tools openssh-server
```

## 2. Create a user with sudo privileges

```
adduser thorgan  
usermod -aG wheel thorgan  
passwd thorgan
```

## 3. Start and enable `sshd` to accept SSH connections

```
service sshd start && systemctl enable sshd
```