

ArchLinux - Installation

vendredi 6 janvier 2023 08:10

Formateur : Loup FORMENT

Lien important pour l'installation :

https://wiki.archlinux.org/title/installation_guide (lien anglais)

[https://wiki.archlinux.org/title/Installation_guide_\(Fran%C3%A7ais\)](https://wiki.archlinux.org/title/Installation_guide_(Fran%C3%A7ais)) (lien français)

<https://www.reddit.com/r/unixporn/> (Liste d'interfaces graphiques)

<https://gist.github.com/mjnaderi/28264ce68f87f52f2cabb823a503e673> (LVM + encrypt)

<https://gist.github.com/NickMcSweeney/0549c0ebbd069aa5b474bad3051d5cc3>

Récapitulatif des commandes tapé pour installer Archlinux (sans le mode faci... Archinstall !)

Prérogative :

1ère installation : une partition /

Dès le boot : le clavier est configuré en qwerty

Pour voir la liste des clavier disponible :

1.5 Disposition du clavier en console

La disposition par défaut est le clavier **US**. Les dispositions disponibles peuvent être listées par :

```
# ls /usr/share/kbd/keymaps/**/*.map.gz
```

```
/usr/share/kbd/keymaps/mac/all/mac-fr.map.gz
/usr/share/kbd/keymaps/mac/all/mac-fr_CH- latin1.map.gz
/usr/share/kbd/keymaps/mac/all/mac-it.map.gz
/usr/share/kbd/keymaps/mac/all/mac-no- latin1.map.gz
/usr/share/kbd/keymaps/mac/all/mac-pl.map.gz
/usr/share/kbd/keymaps/mac/all/mac-pt- latin1.map.gz
/usr/share/kbd/keymaps/mac/all/mac-se.map.gz
/usr/share/kbd/keymaps/mac/all/mac-template.map.gz
/usr/share/kbd/keymaps/mac/all/mac-uk.map.gz
/usr/share/kbd/keymaps/mac/all/mac-us.map.gz
/usr/share/kbd/keymaps/mac/include/mac-euro.map.gz
/usr/share/kbd/keymaps/mac/include/mac-euro2.map.gz
/usr/share/kbd/keymaps/pine/en.map.gz
/usr/share/kbd/keymaps/sun/sundvorak.map.gz
/usr/share/kbd/keymaps/sun/sunkeymap.map.gz
/usr/share/kbd/keymaps/sun/sunt4-es.map.gz
/usr/share/kbd/keymaps/sun/sunt4-fi- latin1.map.gz
/usr/share/kbd/keymaps/sun/sunt4-no- latin1.map.gz
/usr/share/kbd/keymaps/sun/sunt5-cz-us.map.gz
/usr/share/kbd/keymaps/sun/sunt5-de- latin1.map.gz
/usr/share/kbd/keymaps/sun/sunt5-es.map.gz
/usr/share/kbd/keymaps/sun/sunt5-fi- latin1.map.gz
/usr/share/kbd/keymaps/sun/sunt5-fr- latin1.map.gz
/usr/share/kbd/keymaps/sun/sunt5-ru.map.gz
/usr/share/kbd/keymaps/sun/sunt5-uk.map.gz
/usr/share/kbd/keymaps/sun/sunt5-us-cz.map.gz
/usr/share/kbd/keymaps/sun/sunt6-uk.map.gz
/usr/share/kbd/keymaps/sun/sun-pl.map.gz
/usr/share/kbd/keymaps/sun/sun-pl-altgraph.map.gz
root@archiso ~ #
```

Pour passer en AZERTY :

Pour modifier la disposition du **clavier**, passez le nom du fichier correspondant à **loadkeys(1)** en omettant le chemin du fichier ainsi que l'extension. Par exemple pour la disposition **Française** vous pouvez utiliser :

```
# loadkeys fr- latin1
```

```
root@archiso ~ # loadkeys fr- latin1
root@archiso ~ # azertu
```

Pour connaître les périphériques réseau :

1.7 Connexion à Internet

Pour configurer la connexion réseau dans l'image *live*, suivez ces étapes :

- Vérifiez que votre [Carte réseau](#) est répertoriée et activée, par exemple avec `ip-link(8)`:

```
# ip link
```

```
root@archiso ~ # ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
    link/ether 00:15:5d:19:3b:02 brd ff:ff:ff:ff:ff:ff
```

- Pour les connexions sans fil comme le *Wi-Fi* ou [les réseaux mobiles \(4G...\)](#), vérifiez que l'utilitaire `rftkill` ne bloque pas l'interface.
- Connexion au réseau:
 - Ethernet—Connectez le câble.
 - Wi-Fi—Authentifiez-vous sur le réseau sans fil avec `iwctl`.
 - Réseaux mobiles—Connectez-vous au réseau avec l'utilitaire `mmcli`.
- Configurez votre connexion réseau:
 - **DHCP**: l'adressage dynamique des adresses IP et des serveurs DNS (fourni par [systemd-networkd \(Français\)](#) et [systemd-resolved \(Français\)](#) devraient fonctionner par défaut) pour les protocoles [Ethernet](#), [Wi-Fi](#) and [Mobiles](#).
 - Adresse IP statique : suivez [Network configuration#Static IP address](#).
- La connexion peut être confirmée avec `ping`:

```
# ping archlinux.org
```

```
root@archiso ~ # ping archlinux.org
PING archlinux.org (95.217.163.246) 56(84) bytes of data.
64 bytes from archlinux.org (95.217.163.246): icmp_seq=1 ttl=49 time=44.2 ms
64 bytes from archlinux.org (95.217.163.246): icmp_seq=2 ttl=49 time=42.5 ms
64 bytes from archlinux.org (95.217.163.246): icmp_seq=3 ttl=49 time=43.4 ms
64 bytes from archlinux.org (95.217.163.246): icmp_seq=4 ttl=49 time=42.4 ms
64 bytes from archlinux.org (95.217.163.246): icmp_seq=5 ttl=49 time=43.1 ms
64 bytes from archlinux.org (95.217.163.246): icmp_seq=6 ttl=49 time=43.9 ms
^C
--- archlinux.org ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5007ms
rtt min/avg/max/mdev = 42.355/43.227/44.186/0.671 ms
```

1.8 Update the system clock

In the live environment `systemd-timesyncd` is enabled by default and time will be synced automatically once a connection to the internet is established.

Use `timedatectl(1)` to ensure the system clock is accurate:

```
# timedatectl status
```

```
root@archiso ~ # timedatectl status
          Local time: Fri 2023-01-06 08:41:22 UTC
          Universal time: Fri 2023-01-06 08:41:22 UTC
                RTC time: Fri 2023-01-06 08:41:22
                Time zone: UTC (UTC, +0000)
System clock synchronized: yes
              NTP service: active
```

RTC in local TZ: no

1.9 Partition the disks

When recognized by the live system, disks are assigned to a **block device** such as `/dev/sda`, `/dev/nvme0n1` or `/dev/mmcblk0`. To identify these devices, use **lsblk** or **fdisk**.

```
# fdisk -l
```

```
root@archiso ~ # lsblk
NAME MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0  7:0    0 706.5M  1 loop /run/archiso/airootfs
sda     8:0    0  127G   0 disk
sr0     11:0   1 818.3M   0 rom
```

Ici, je sais que mon disque SDA est le volume sur lequel je vais devoir travailler

1.9 Partitionnement des disques

Une fois reconnus par le système *live*, les disques se verront affectés un **périphérique de type bloc** tel que `/dev/sda`, `/dev/nvme0n1` ou `/dev/mmcblk0`. Pour identifier ces périphériques, utilisez **lsblk** ou **fdisk**.

```
# fdisk -l
```

Les résultats se terminant par: `rom`, `loop` ou `airoot` peuvent être ignorés.

Les **partitions** suivantes sont **nécessaires** sur un périphériques choisi:

- Une partition **racine** `/`.
- Pour démarrer en mode **UEFI**: une **partition EFI**.

Si vous souhaitez utiliser **LVM**, un **chiffrement de votre système** ou encore **RAID**, faites le maintenant.

- Utilisez **fdisk** or **parted** pour modifier la table de partitions. Par exemple :

```
# fdisk /dev/disque_en_question
```

Note:

- Si votre disque n'apparaît pas, **vérifiez que le contrôleur du disque n'est pas en mode RAID**.
- Si le disque depuis lequel vous souhaitez démarrer **possède déjà une partition EFI**, n'en recréez pas une autre mais utilisez la partition pré-existante.
- La **Swap** peut être définie sur un **fichier d'échange** pour les systèmes de fichier qui le prennent en charge.

Note : **cfdisk** fonctionne également



Device	Start	End	Sectors	Size	Type
>> /dev/sda1	2048	266336255	266334208	127G	Linux filesystem

```
Partition UUID: B662C81E-F4C0-4D42-BD7B-7BEA02F8BE69
Partition type: Linux filesystem (0FC63DAF-8483-4772-8E79-3D69D8477DE4)
```

```
[ Delete ] [ Resize ] [ Quit ] [ Type ] [ Help ] [ Write ] [ Dump ]
```

```
root@archiso ~ # lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
loop0 7:0 0 706.5M 1 loop /run/archiso/airootfs
sda 8:0 0 127G 0 disk
└─sda1 8:1 0 127G 0 part
sr0 11:0 1 818.3M 0 rom
```

Avec LVM

Vgcreate + vgcreate + verification avec vgdisplay :

```
root@archiso ~ # pvcreate /dev/sda
Physical volume "/dev/sda" successfully created.
root@archiso ~ # lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
loop0 7:0 0 706.5M 1 loop /run/archiso/airootfs
sda 8:0 0 127G 0 disk
sr0 11:0 1 818.3M 0 rom
root@archiso ~ # vgcreate Arch_System /dev/sda
Volume group "Arch_System" successfully created
root@archiso ~ # lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
loop0 7:0 0 706.5M 1 loop /run/archiso/airootfs
sda 8:0 0 127G 0 disk
sr0 11:0 1 818.3M 0 rom
root@archiso ~ # vgdisplay
--- Volume group ---
VG Name Arch_System
System ID
Format lvm2
Metadata Areas 1
Metadata Sequence No 1
VG Access read/write
VG Status resizable
MAX LV 0
Cur LV 0
Open LV 0
Max PV 0
Cur PV 1
Act PV 1
VG Size <127.00 GiB
PE Size 4.00 MiB
Total PE 32511
Alloc PE / Size 0 / 0
Free PE / Size 32511 / <127.00 GiB
VG UUID Vn2UvS-ie60-w0cz-dE6R-f7tz-eOtN-LKLweb
root@archiso ~ #
```

Si on doit crypter un point de montage : cryptsetup

```
root@archiso ~ # cryptsetup luksFormat /dev/sda2
WARNING: Device /dev/sda2 already contains a 'crypto_LUKS' superblock signature.

WARNING!
=====
This will overwrite data on /dev/sda2 irrevocably.
```

```
Are you sure? (Type 'yes' in capital letters): YES
Enter passphrase for /dev/sda2:
Verify passphrase:
cryptsetup luksFormat /dev/sda2 6.31s user 0.25s system 44% cpu 14.894 total
root@archiso ~ # cryptsetup open /dev/sda2 /crypt
Enter passphrase for /dev/sda2:
Name "/crypt" invalid. It contains "/".
1 root@archiso ~ # cryptsetup open /dev/sda2 crypt
Enter passphrase for /dev/sda2:
root@archiso ~ # lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINTS
loop0        7:0      0  706.5M 1 loop  /run/archiso/airootfs
sda          8:0      0   127G 0 disk
├─sda1       8:1      0    1G 0 part
├─sda2       8:2      0   126G 0 part
└─crypt      253:0    0   126G 0 crypt
sr0         11:0    1  818.3M 0 rom
root@archiso ~ #
```

```
4 1SDIR
5 sudo cryptsetup luksFormat /dev/sda2
6 cryptsetup luksFormat /dev/sda2
```

1.9.1 Exemples de Partitionnement

UEFI avec GPT

Point de Montage	Partition	Type de partition	Taille suggérée
/mnt/boot ¹	/dev/efi_system_partition	EFI system partition	Au moins 300 MiB
[SWAP]	/dev/partition_d'échange	Linux swap	Plus de 512 MiB
/mnt	/dev/partition_racine	Linux x86-64 root (/)	Le reste du disque

1. D'autres points de montage, tels que `/mnt/efi` ne devrait être envisagé que si le chargeur d'amorçage utilisé est capable de charger le **noyau** et l'**initramfs** directement depuis la partition racine. Voir l'avertissement dans [Arch boot process \(Français\)#Chargeur d'amorçage](#).

BIOS avec MBR

Point de Montage	Partition	Type de Partition	Taille suggérée
[SWAP]	/dev/partition_d'échange	Linux swap	Plus de 512 MiB
/mnt	/dev/partition_racine	Linux	Le reste du disque

Pour d'autres exemples: [Partitioning#Example layouts](#).

1.10 Formatage des partitions

Une fois les partitions créées, celles-ci doivent être formatées avec un **système de fichier** approprié. Consultez [File systems#Create a file system](#) pour plus de détails.

Par exemple, pour créer un système de fichier `ext4` sur `/dev/partition_racine`, utilisez :

```
# mkfs.ext4 /dev/partition_racine
```

Si vous avez créé une partition **d'échange**, initialisez-la avec `mkswap(8)`:

```
# mkswap /dev/partition_d'échange
```

Note: Pour les configurations avec LVM, chiffrement ou RAID, remplacez `/dev/*_partition` par le chemin vers le périphérique de type bloc approprié.

Si vous avez créé une partition système EFI, **formatez-la** en FAT32 avec `mkfs.fat(8)`.

Attention: Ne formatez la partition système EFI que si vous l'avez créée pendant le partitionnement. S'il y avait déjà une partition système EFI sur le disque précédemment, son formatage peut détruire les chargeurs d'amorçage des autres systèmes d'exploitation installés.

```
# mkfs.fat -F 32 /dev/efi_system_partition
```

Pour UEFI -> toujours `mkfs.fat -F 32`

```
sr0         11:0    1  818.3M 0 rom
root@archiso ~ # umount /dev/sda1
root@archiso ~ # mkfs.fat -F 32 /dev/sda1
mkfs.fat 4.2 (2021-01-31)
```

```
root@archiso ~ # mkfs.ext4 /dev/sda1
mke2fs 1.46.5 (30-Dec-2021)
Discarding device blocks: done
Creating filesystem with 33292032 4k blocks and 8323072 inodes
```

```
Filesystem UUID: c9edad5b-28a4-469a-b6d3-f71857ba5ef4
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
    4096000, 7962624, 11239424, 20480000, 23887872

Allocating group tables: done
Writing inode tables: done
Creating journal (131072 blocks): done
Writing superblocks and filesystem accounting information: done
```

Warning : si jamais on rentre cette commande :

```
root@archiso ~ # mkfs.ext4 /dev/sda_
```

Il efface la partition créer, et cela peut aussi endommager les points d'amorçages, faire très attention

```
root@archiso ~ # lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
loop0 7:0 0 706.5M 1 loop /run/archiso/airootfs
sda 8:0 0 127G 0 disk
└─sda1 8:1 0 127G 0 part
sr0 11:0 1 818.3M 0 rom

root@archiso ~ # mkfs.ext4 /dev/sda1
mke2fs 1.46.5 (30-Dec-2021)
Discarding device blocks: done
Creating filesystem with 33291776 4k blocks and 8323072 inodes
Filesystem UUID: d1191cf6-7938-499f-adb3-54850095f5c0
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
    4096000, 7962624, 11239424, 20480000, 23887872

Allocating group tables: done
Writing inode tables: done
Creating journal (131072 blocks): done
Writing superblocks and filesystem accounting information: done

root@archiso ~ # lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
loop0 7:0 0 706.5M 1 loop /run/archiso/airootfs
sda 8:0 0 127G 0 disk
└─sda1 8:1 0 127G 0 part
sr0 11:0 1 818.3M 0 rom
```

1.11 Montage des systèmes de fichiers

Montez la partition racine dans `/mnt`. Par exemple, si la partition racine est `/dev/partition_racine` :

```
# mount /dev/partition_racine /mnt
```

Créez tous les points de montage restants (tels que `/mnt/efi`) et montez les volumes correspondants.

Astuce: Utilisez `mount(8)` avec l'option `--mkdir` pour créer le point de montage spécifié. Vous pouvez également le créer au préalable à l'aide de `mkdir(1)`.

Pour les systèmes avec un UEFI, montez la partition système EFI :

```
# mount --mkdir /dev/efi_system_partition /mnt/boot
```

Si vous avez créé une partition **d'échange**, activez-la avec `swapon(8)` :

```
# swapon /dev/partition_d'échange
```

`genfstab(8)` détectera plus tard les systèmes de fichiers et l'espace d'échange montés.

Respecter ordre de point de montage :

Racine en 1er
Boot en second
Home en dernier

But du jeu de arch -> installer racine dans mnt

Pour monter à la racine :

Pour monter dans la racine en créant un boot :

```
32 root@archiso ~ # mount --mkdir /dev/sda1 /mnt/boot
root@archiso ~ # lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
loop0 7:0 0 706.5M 1 loop /run/archiso/airootfs
sda 8:0 0 127G 0 disk
└─sda1 8:1 0 1G 0 part /mnt/boot
```

```
└─sda2  8:2   0   50G  0 part /mnt
└─sda3  8:3   0   76G  0 part
sr0     11:0   1 818.3M  0 rom
root@archiso ~ #
```

Pour monter une partition home :

```
25 ls /mnt/
26 mount --mkdir /dev/sda3 /mnt/home
27 lsblk
```

```
root@archiso ~ # lsblk
NAME MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0  7:0    0 706.5M  1 loop /run/archiso/airootfs
sda     8:0    0  127G  0 disk
└─sda1  8:1    0  127G  0 part
sr0     11:0   1 818.3M  0 rom
root@archiso ~ # mount /dev/sda1 /mnt
root@archiso ~ # lsblk
NAME MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0  7:0    0 706.5M  1 loop /run/archiso/airootfs
sda     8:0    0  127G  0 disk
└─sda1  8:1    0  127G  0 part /mnt
sr0     11:0   1 818.3M  0 rom
```

Si besoin, pour **umount** :

```
root@archiso ~ # umount /dev/sda1
root@archiso ~ # lsblk
NAME MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0  7:0    0 706.5M  1 loop /run/archiso/airootfs
sda     8:0    0  127G  0 disk
└─sda1  8:1    0  127G  0 part
sr0     11:0   1 818.3M  0 rom
```

Pour effacer la partition :

```
root@archiso ~ # dd if=/dev/zero of=/dev/sda bs=1100 count=1
1+0 records in
1+0 records out
1100 bytes (1.1 kB, 1.1 KiB) copied, 0.0270145 s, 40.7 kB/s
root@archiso ~ # lsblk
NAME MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0  7:0    0 706.5M  1 loop /run/archiso/airootfs
sda     8:0    0  127G  0 disk
sr0     11:0   1 818.3M  0 rom
```

On peut utiliser **fdisk** :

```
Command (m for help): m

Help:

DOS (MBR)
 a  toggle a bootable flag
 b  edit nested BSD disklabel
 c  toggle the dos compatibility flag

Generic
 d  delete a partition
 F  list free unpartitioned space
 l  list known partition types
 n  add a new partition
 p  print the partition table
 t  change a partition type
 v  verify the partition table
 i  print information about a partition

Misc
 m  print this menu
 u  change display/entry units
 x  extra functionality (experts only)

Script
 I  load disk layout from sfdisk script file
 O  dump disk layout to sfdisk script file

Save & Exit
```

```

w write table to disk and exit
q quit without saving changes

Create a new label
g create a new empty GPT partition table
G create a new empty SGI (IRIX) partition table
o create a new empty DOS partition table
s create a new empty Sun partition table

Command (m for help): o
Created a new DOS disklabel with disk identifier 0x212b9edf.

```

On arrive dans un prompt qui nous permet de sélectionner les options désiré : par exemple o pour créer un nouveau label : **o pour passer en dos**

```

Command (m for help): o
Created a new DOS disklabel with disk identifier 0x212b9edf.

```

```

Command (m for help): n
Partition type
  p primary (0 primary, 0 extended, 4 free)
  e extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-266338303, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-266338303, default 266338303):

Created a new partition 1 of type 'Linux' and of size 127 GiB.

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

```

Puis **n** pour une nouvelle partition, **p** pour primary

On peut sélectionner le nombre de partition

W pour write les modifications

Résultat des modification :

```

root@archiso ~ # lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
loop0 7:0 0 706.5M 1 loop /run/archiso/airootfs
sda 8:0 0 127G 0 disk
└─sda1 8:1 0 127G 0 part
sr0 11:0 1 818.3M 0 rom

```

Je suis bien repassé en DOS sur mon disque pour l'installation legacy

Je dois juste lui redonner son système de fichier + le mount :

Respecter ordre de point de montage :

Racine en 1er

Boot en second

Home en dernier

```

root@archiso ~ # mkfs.ext4 /dev/sda1
mke2fs 1.46.5 (30-Dec-2021)
Creating device blocks: done
Creating filesystem with 33292032 4k blocks and 8323072 inodes
Filesystem UUID: c9edad5b-28a4-469a-b6d3-f71857ba5ef4
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
    4096000, 7962624, 11239424, 20480000, 23887872

Allocating group tables: done
Writing inode tables: done
Creating journal (131072 blocks): done
Writing superblocks and filesystem accounting information: done

root@archiso ~ # lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
loop0 7:0 0 706.5M 1 loop /run/archiso/airootfs
sda 8:0 0 127G 0 disk
└─sda1 8:1 0 127G 0 part
sr0 11:0 1 818.3M 0 rom
root@archiso ~ # mount /dev/sda1 /mnt
root@archiso ~ # lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
loop0 7:0 0 706.5M 1 loop /run/archiso/airootfs

```



```
sda      8:0    0   127G  0 disk
└─sda1   8:1    0   127G  0 part /mnt
sr0      11:0   1 818.3M  0 rom
```

2 Installation

2.1 Sélection du miroir

Les paquets à installer doivent être téléchargés depuis les [miroirs](#) des dépôts officiels tels que définis dans `/etc/pacman.d/mirrorlist`. Sur le système *live*, après connexion à Internet, [reflector](#) met à jour la liste des miroirs en sélectionnant les 20 miroirs les plus récemment synchronisés et en les triant par vitesse de téléchargement.

Plus le miroir se trouve haut dans la liste, plus grande est sa priorité lors d'un téléchargement. Vérifiez le contenu de `/etc/pacman.d/mirrorlist` et modifiez le si besoin. Placez les miroirs les plus proches géographiquement en haut de la liste, bien que d'autres critères sont à prendre en compte.

Par exemple pour trouver un [miroir](#) plus proche de chez vous (et/ou plus rapide) en utilisant [reflector](#) :

```
# reflector --country France --age 12 --protocol https --sort rate --save /etc/pacman.d/mirrorlist
```

Cette commande devrait sélectionner les miroirs HTTPS synchronisés ces 12 dernières heures en France, les trier par vitesse de téléchargement, et mettre à jour le fichier `/etc/pacman.d/mirrorlist`.

`pacstrap` copiera plus tard ce fichier sur le nouveau système, prendre votre temps sur cette étape peut donc vous en faire gagner plus tard.

2.2 Install essential packages

Use the `pacstrap(8)` script to install the [base](#) package, Linux [kernel](#) and firmware for common hardware:

```
# pacstrap -K /mnt base linux linux-firmware
```

Tip:

- You can substitute [linux](#) for a [kernel](#) package of your choice, or you could omit it entirely when installing in a [container](#).
- You could omit the installation of the firmware package when installing in a virtual machine or container.

The [base](#) package does not include all tools from the live installation, so installing other packages may be necessary for a fully functional base system. In particular, consider installing:

- userspace utilities for the management of [file systems](#) that will be used on the system,
- utilities for accessing [RAID](#) or [LVM](#) partitions,
- specific firmware for other devices not included in [linux-firmware](#) (e.g. [sof-firmware](#) for [sound cards](#)),
- software necessary for [networking](#) (e.g. a network manager or DHCP client),
- a [text editor](#),
- packages for accessing documentation in [man](#) and [info](#) pages: [man-db](#), [man-pages](#) and [texinfo](#).

To [install](#) other packages or package groups, append the names to the `pacstrap` command above (space separated) or use `pacman` while [chrooted into the new system](#). For comparison, packages available in the live system can be found in [pkglist.x86_64.txt](#).

```
bzip2-1.0.8-5-x86_64 58.4 KiB 3.35 MiB/s 00:00 [#####] 100%
json-c-0.16-1-x86_64 54.6 KiB 3.14 MiB/s 00:00 [#####] 100%
jansson-2.14-2-x86_64 51.7 KiB 3.16 MiB/s 00:00 [#####] 100%
libnetfilter_conntrack-1.0.9-1-x86_64 49.0 KiB 2.99 MiB/s 00:00 [#####] 100%
mkinitcpio-34-2-any 47.4 KiB 2.72 MiB/s 00:00 [#####] 100%
libsysprof-capture-3.46.0-1-x86_64 45.5 KiB 3.17 MiB/s 00:00 [#####] 100%
libffi-3.4.4-1-x86_64 45.3 KiB 2.60 MiB/s 00:00 [#####] 100%
libcap-ng-0.8.3-1-x86_64 41.6 KiB 2.39 MiB/s 00:00 [#####] 100%
argon2-20190702-4-x86_64 32.4 KiB 2028 KiB/s 00:00 [#####] 100%
linux-firmware-uhence-20221214.f3c283e-1-any 29.7 KiB 2.23 MiB/s 00:00 [#####] 100%
npth-1.6-3-x86_64 24.5 KiB 1441 KiB/s 00:00 [#####] 100%
libuio-0.3.2-4-x86_64 18.4 KiB 1314 KiB/s 00:00 [#####] 100%
libnftnl-1.0.2-1-x86_64 17.1 KiB 1318 KiB/s 00:00 [#####] 100%
filesystem-2022.10.18-1-x86_64 14.4 KiB 899 KiB/s 00:00 [#####] 100%
libmnl-1.0.5-1-x86_64 11.2 KiB 863 KiB/s 00:00 [#####] 100%
ca-certificates-utils-20220905-1-any 10.3 KiB 794 KiB/s 00:00 [#####] 100%
pacman-mirrorlist-20221204-1-any 7.0 KiB 459 KiB/s 00:00 [#####] 100%
systemd-sysucompat-252.4-2-x86_64 6.1 KiB 358 KiB/s 00:00 [#####] 100%
pambase-20221020-1-any 3.1 KiB 181 KiB/s 00:00 [#####] 100%
base-3-1-any 2.2 KiB 172 KiB/s 00:00 [#####] 100%
ca-certificates-20220905-1-any 2.0 KiB 127 KiB/s 00:00 [#####] 100%
```

```

Total (121/121) 460.0 MiB 23.8 MiB/s 00:19 [#####] 100%
(121/121) checking keys in keyring [#####] 100%
(121/121) checking package integrity [#####] 100%
error: openssl: signature from "Pierre Schmitz <pierre@archlinux.org>" is marginal trust
:: File /mnt/var/cache/pacman/pkg/openssl-3.0.7-4-x86_64.pkg.tar.zst is corrupted (invalid or corrupted package (PGP signature))
.
Do you want to delete it? [Y/n]
error: failed to commit transaction (invalid or corrupted package)
Errors occurred, no packages were upgraded.
=> ERROR: Failed to install packages to new root
pacstrap -K /mnt base linux linux-firmware 3.15s user 3.37s system 22% cpu 29.323 total

```

On constate un problème au niveau des signatures : ces commandes peuvent régler le problème :

```

74 pacman -S archlinux-keyring
75 pacman -Sy
76 pacman -S archlinux-keyring
77 pacstrap /mnt base linux linux-firmware

```

```

root@archiso ~ # pacman -Sy
:: Synchronizing package databases...
core 152.1 KiB 634 KiB/s 00:00 [#####] 100%
extra 1748.0 KiB 21.3 MiB/s 00:00 [#####] 100%
community 7.2 MiB 10.0 MiB/s 00:01 [#####] 100%
root@archiso ~ # pacman -S archlinux-keyring
resolving dependencies...
looking for conflicting packages...

Packages (1) archlinux-keyring-20221220-1

Total Download Size: 1.13 MiB
Total Installed Size: 1.60 MiB
Net Upgrade Size: 0.02 MiB

:: Proceed with installation? [Y/n] y
:: Retrieving packages...
archlinux-keyring-20221220-1-any 1153.7 KiB 11.3 MiB/s 00:00 [#####] 100%
(1/1) checking keys in keyring [#####] 100%
(1/1) checking package integrity [#####] 100%
(1/1) loading package files [#####] 100%
(1/1) checking for file conflicts [#####] 100%
(1/1) checking available disk space [#####] 100%
:: Processing package changes...
(1/1) upgrading archlinux-keyring [#####] 100%
=> Appending keys from archlinux.gpg...
=> Disabling revoked keys in keyring...
-> Disabled 4 keys.
=> Updating trust database...
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: depth: 0 valid: 1 signed: 6 trust: 0-, 0q, 0n, 0n, 0f, 1u
gpg: depth: 1 valid: 6 signed: 95 trust: 0-, 0q, 0n, 0n, 0f, 0u
gpg: depth: 2 valid: 75 signed: 26 trust: 75-, 0q, 0n, 0n, 0f, 0u
gpg: next trustdb check due at 2023-04-21
=> Updating trust database...
gpg: next trustdb check due at 2023-04-21
:: Running post-transaction hooks...
(1/2) Reloading system manager configuration...
(2/2) Arming ConditionNeedsUpdate...
root@archiso ~ #

```

Pacman -Sy permet de rafraichir les signatures, on repasse ensuite la commande **pacman -S archlinux-keyring**

Puis **pacstrap /mnt base linux linux-firmware**

3 Configure the system

3.1 Fstab

Generate an **fstab** file (use **-U** or **-L** to define by **UUID** or labels, respectively):

```
# genfstab -U /mnt >> /mnt/etc/fstab
```

Check the resulting **/mnt/etc/fstab** file, and **edit** it in case of errors.

```
1 root@archiso ~ # genfstab -U /mnt >> /mnt/etc/fstab
```

(à faire en dernier avant le reboot pour ne pas se couper la main si jamais on doit refaire des modifications avec les disques)

Installation internet : **pacstrap -K /mnt networkmanager**

```

Skipped: Running in chroot.
root@archiso ~ # pacstrap -K /mnt networkmanager

```

(AVANT LE CHROOT)

3.2 Chroot

Change root into the new system:

```
# arch-chroot /mnt
```

```
# arch-chroot /mnt
```

```
root@archiso ~ # arch-chroot /mnt  
[root@archiso /]#
```

3.3 Fuseau Horaire

Définissez le [fuseau horaire](#), par exemple pour la France:

```
# ln -sf /usr/share/zoneinfo/Europe/Paris /etc/localtime
```

Lancez [hwclock\(8\)](#) pour générer `/etc/adjtime` :

```
# hwclock --systohc
```

Cette commande repose sur l'hypothèse que l'horloge matérielle est réglée sur [UTC](#). Consultez [System time \(Français\)#Standard de temps](#) pour plus de détails.

```
[root@archiso /]# ln -sr /usr/share/zoneinfo/Europe/Paris /etc/localtime  
[root@archiso /]# hwclock --systohc  
[root@archiso /]#
```

Pour vérifier la configuration : commande `time`

```
[root@archiso /]# time  
  
real    0m0.000s  
user    0m0.000s  
sys     0m0.000s  
[root@archiso /]# date  
Fri Jan  6 11:27:26 CET 2023
```

3.4 Localization

[Edit](#) `/etc/locale.gen` and uncomment `en_US.UTF-8 UTF-8` and other needed [locales](#). Generate the locales by running:

```
# locale-gen
```

[Create](#) the `locale.conf(5)` file, and [set the LANG variable](#) accordingly:

```
/etc/locale.conf
```

```
LANG=en_US.UTF-8
```

If you [set the console keyboard layout](#), make the changes persistent in `vconsole.conf(5)`:

```
/etc/vconsole.conf
```

```
KEYMAP=de-Latin1
```

Pour France, décommenter `fr_FR.UTF-8 UTF-8` :

```
#fr_CA.UTF-8 UTF-8  
#fr_CA ISO-8859-1  
#fr_CH.UTF-8 UTF-8  
#fr_CH ISO-8859-1  
fr_FR.UTF-8 UTF-8  
#fr_FR ISO-8859-1  
#fr_FR@euro ISO-8859-15  
#fr_LU.UTF-8 UTF-8
```

On créer le fichier locale.conf :

```
[root@archiso ~]# touch /etc/locale.conf
[root@archiso ~]# ls /etc/
X11                  default              gshadow              krb5.conf            makepkg.conf         openldap             protocols            skel
adjtime              depmod.d             gshadow-             ld.so.cache          mke2fs.conf          os-release           request-key.conf     ssl
arch-release         e2scrub.conf         host.conf            ld.so.conf           mkinitcpio.conf      pacman.conf          request-key.d        subgid
audit                environment          hosts                ld.so.conf.d         mkinitcpio.d         pacman.d             resolv.conf          subuid
bash.bash_logout     ethertypes           initcpio             libaudit.conf        modprobe.d           pam.d               rpc                  systcl.d
bash.bashrc          fstab                inputrc             libn1                 modules-load.d       passwd              securetty            systemd
bindresuport.blacklist fstab.pacnew          iproute2            locale.conf          ntab                 passwd-             security            tmpfiles.d
binfmt.d             gai.conf             iptables            locale.gen            nanorc               pinentry            services             tpm2-tss
ca-certificates      gprofng.rc           issue               localtime            netconfig            pkcs11              shadow              udev
crypttab             group               kernel              login.defs            nscd.conf            profile             shadow-             xattr.conf
debuginfod           group-              keyutils            machine-id            nsswitch.conf        profile.d            shells              xdg
```

Il faut crée ensuite le vconsole.conf

If you [set the console keyboard layout](#), make the changes persistent in vconsole.conf(5):

```
/etc/vconsole.conf

KEYMAP=de-Latin1
```

3.5 Network configuration

Create the [hostname](#) file:

```
/etc/hostname

myhostname
```

Complete the [network configuration](#) for the newly installed environment. That may include installing suitable [network management](#) software.

Host name = nom de la machine

```
[root@archiso ~]# cat /etc/hostname
ArchBetterThenWindows
[root@archiso ~]# S
```

Optionnel mais à faire pour le LVM et le cryptage

3.6 Initramfs

La création d'un nouvel *initramfs* n'est généralement pas nécessaire, car [mkinitcpio](#) a été lancé lors de l'installation du [noyau](#) avec *pacstrap*.

Pour [LVM](#), le [chiffrement](#) ou [RAID](#), modifiez [mkinitcpio.conf\(5\)](#) et recréez l'image *initramfs*:

```
# mkinitcpio -P
```

[mkinitcpio](#) est un script Bash utilisé pour créer un environnement «[ramdisk](#)» [initial](#). Extrait de la page de manuel [mkinitcpio\(8\)](#) :

Le «[ramdisk](#)» [initial](#) est par essence un très petit environnement (early userspace) qui charge divers modules du noyau et configure les choses nécessaires avant de céder le contrôle à `init`. Cela permet d'avoir, par exemple, des systèmes de fichiers racine chiffrés et des systèmes de fichiers racine sur une matrice RAID logicielle. *mkinitcpio* permet une extension facile avec des «[hooks](#)» personnalisés, a une autodétection à l'exécution, et beaucoup d'autres fonctionnalités.

[https://wiki.archlinux.org/title/Mkinitcpio_\(Fran%C3%A7ais\)](https://wiki.archlinux.org/title/Mkinitcpio_(Fran%C3%A7ais))

1 Installation

[Installez](#) le paquet [mkinitcpio](#), qui est une dépendance du paquet [linux](#), donc la plupart des utilisateurs l'auront déjà installé.

Les utilisateurs avancés peuvent souhaiter installer la dernière version de développement de *mkinitcpio* depuis Git avec le paquet [mkinitcpio-git](#)^{AUR}.

Note: Il est **tortement** recommande de suivre la [mailing list arch-projects](#) si vous utilisez *mkinitcpio* depuis Git!

2 Création et activation d'images

2.1 Génération automatique

Chaque fois qu'un noyau est installé ou mis à jour, un «hook» de **pacman** génère automatiquement un fichier *.preset* enregistré dans `/etc/mkinitcpio.d/`. Par exemple `linux.preset` pour le paquet stable officiel **linux** du noyau. Un preset est simplement une liste d'informations requises pour créer des images ramdisk initiales, au lieu de spécifier manuellement les différents paramètres et l'emplacement des fichiers de sortie. Par défaut, il contient les instructions pour créer deux images :

1. l'image ramdisk *par défaut* créée suivant les directives spécifiées dans la **#Configuration** de *mkinitcpio*, et
2. l'image ramdisk *fallback*, identique à la précédente sauf que le hook *autodetect* est ignoré pendant la création, incluant ainsi une gamme complète de modules qui prends en charge la plupart des systèmes.

Après avoir créé le preset, le hook *pacman* appelle le script *mkinitcpio* qui génère les deux images, en utilisant les informations fournies dans le preset.

Note: Les fichiers *preset* sont utilisés pour régénérer automatiquement les *initramfs* après une mise à jour du noyau ; soyez prudent lorsque vous les modifiez.

2.2 Génération manuelle

Pour exécuter le script manuellement, référez-vous à la page de manuel [mkinitcpio\(8\)](#) pour les instructions. En particulier, pour (re-)générer le préréglage fourni par un paquet du noyau, utilisez l'option `-p / --preset` suivie du préréglage à utiliser. Par exemple, pour le paquet **linux**, utilisez la commande :

```
# mkinitcpio -p linux
```

Pour (re)générer tous les préréglages existants, utilisez le paramètre `-P / --allpresets`. Ceci est typiquement utilisé pour régénérer toutes les images *initramfs* après un changement de la **#Configuration** globale :

```
# mkinitcpio -P
```

Les utilisateurs peuvent créer un nombre arbitraire d'images *initramfs* avec une variété de configurations différentes. L'image désirée doit être spécifiée dans le fichier de configuration du chargeur d'amorçage respectif.

https://wiki.archlinux.org/title/dm-crypt/Encrypting_an_entire_system

```
[root@archiso ~/]# mkinitcpio -p linux
==> Building image from preset: /etc/mkinitcpio.d/linux.preset: 'default'
-> -k /boot/vmlinuz-linux -c /etc/mkinitcpio.conf -g /boot/initramfs-linux.img
==> Starting build: 6.1.4-arch1-1
-> Running build hook: [base]
-> Running build hook: [udev]
-> Running build hook: [autodetect]
-> Running build hook: [modconf]
-> Running build hook: [kms]
-> Running build hook: [keyboard]
-> Running build hook: [keymap]
-> Running build hook: [consolefont]
==> WARNING: consolefont: no font found in configuration
-> Running build hook: [block]
-> Running build hook: [filesystems]
-> Running build hook: [fsck]
==> Generating module dependencies
==> Creating zstd-compressed initcpio image: /boot/initramfs-linux.img
==> Image generation successful
==> Building image from preset: /etc/mkinitcpio.d/linux.preset: 'fallback'
-> -k /boot/vmlinuz-linux -c /etc/mkinitcpio.conf -g /boot/initramfs-linux-fallback.img -S autodetect
==> Starting build: 6.1.4-arch1-1
-> Running build hook: [base]
-> Running build hook: [udev]
-> Running build hook: [modconf]
-> Running build hook: [kms]
==> WARNING: Possibly missing firmware for module: ast
-> Running build hook: [keyboard]
==> WARNING: Possibly missing firmware for module: xhci_pci
-> Running build hook: [keymap]
-> Running build hook: [consolefont]
==> WARNING: consolefont: no font found in configuration
-> Running build hook: [block]
==> WARNING: Possibly missing firmware for module: bfa
==> WARNING: Possibly missing firmware for module: qed
==> WARNING: Possibly missing firmware for module: wd719x
==> WARNING: Possibly missing firmware for module: qla2xxx
==> WARNING: Possibly missing firmware for module: aic94xx
==> WARNING: Possibly missing firmware for module: qla1280
-> Running build hook: [filesystems]
-> Running build hook: [fsck]
==> Generating module dependencies
==> Creating zstd-compressed initcpio image: /boot/initramfs-linux-fallback.img
==> Image generation successful
[root@archiso ~/]#
```

```
[root@archiso ~/]# blkid | grep da2
/dev/sda2: UUID="9a29777d-4f02-44bd-b5bf-62463c70dd4d" TYPE="crypto_LUKS" PARTUUID="b35c71b1-a661-544a-8306-7c64beada0e"
```

3.7 Mot de passe administrateur

Définissez un **mot de passe** pour root:

```
# passwd
```

```
[root@archiso ~]# passwd root
New password:
Retype new password:
passwd: password updated successfully
[root@archiso ~]# _
```

3.8 Boot loader

Choose and install a Linux-capable **boot loader**. If you have an Intel or AMD CPU, enable **microcode** updates in addition.

https://fr.wikipedia.org/wiki/Chargeur_d'amor%C3%A7age

<https://wiki.archlinux.org/title/GRUB>

Chargeur d'amorçage

🌐 28 langues ▾

Article Discussion

Lire Modifier Modifier le code Voir l'historique

Un **chargeur d'amorçage**¹ (ou **bootloader**) est un **logiciel** permettant de lancer un ou plusieurs **systèmes d'exploitation** (**multiboot**), c'est-à-dire qu'il permet d'utiliser plusieurs systèmes, à des moments différents, sur la même machine. Il est généralement lancé immédiatement après le **démarrage** de l'appareil, puis recherche dans le **firmware** de l'ordinateur des informations sur l'emplacement du bootloader. Ce processus est communément appelé « amorçage »².

2.3 Installation

Install the **grub** package. (It will replace **grub-legacy**^{AUR} if that is already installed.) Then do:

```
# grub-install --target=i386-pc /dev/sdX
```

where `i386-pc` is deliberately used regardless of your actual architecture, and `/dev/sdX` is the **disk (not a partition)** where GRUB is to be installed. For example `/dev/sda` or `/dev/nvme0n1`, or `/dev/mmcblk0`. See [Device file#Block device names](#) for a description of the block device naming scheme.

Now you must [generate the main configuration file](#).

If you use **LVM** for your `/boot`, you can install GRUB on multiple physical disks.

Tip: See [/Tips and tricks#Alternative installation methods](#) for other ways to install GRUB, such as to a USB stick.

See [grub-install\(8\)](#) and [GRUB Manual](#)[🔗] for more details on the `grub-install` command.

```
[root@archiso ~]# grub-install --target=igrub-install --target=i386-pc /dev/sda
Installing for i386-pc platform.
Installation finished. No error reported.
```

Générer ensuite le fichier de configuration :

3.1.1 Generate the main configuration file

After the installation, the main configuration file `/boot/grub/grub.cfg` needs to be generated. The generation process can be influenced by a variety of options in `/etc/default/grub` and scripts in `/etc/grub.d/`.

If you have not done additional configuration, the automatic generation will determine the root filesystem of the system to boot for the configuration file. For that to succeed it is important that the system is either booted or chrooted into.

Note:

- The default file path is `/boot/grub/grub.cfg`, not `/boot/grub/i386-pc/grub.cfg`.
- If you are trying to run `grub-mkconfig` in a **chroot** or **systemd-nspawn** container, you might notice that it does not work:
`grub-probe: error: failed to get canonical path of /dev/sdaX`. In this case, try using [arch-chroot](#) as described in the [BBS post](#)[🔗].

Use the `grub-mkconfig` tool to generate `/boot/grub/grub.cfg`:

```
# grub-mkconfig -o /boot/grub/grub.cfg
```

By default the generation scripts automatically add menu entries for all installed Arch Linux **kernels** to the generated configuration.

Tip:

- After installing or removing a **kernel**, you just need to re-run the above `grub-mkconfig` command.
- For tips on managing multiple GRUB entries, for example when using both **linux** and **linux-lts** kernels, see [/Tips and tricks#Multiple entries](#).

To automatically add entries for other installed operating systems, see [#Detecting other operating systems](#).

You can add additional custom menu entries by editing `/etc/grub.d/40_custom` and re-generating `/boot/grub/grub.cfg`. Or you can create `/boot/grub/custom.cfg` and add them there. Changes to `/boot/grub/custom.cfg` do not require re-running `grub-mkconfig`, since `/etc/grub.d/41_custom` adds the necessary `source` statement to the generated configuration file.

Tip: `/etc/grub.d/40_custom` can be used as a template to create `/etc/grub.d/xx_custom`, where `xx` defines the precedence, indicating the

`ip: /etc/grub.d/40_custom` can be used as a template to create `/etc/grub.d/nn_custom`, where `nn` defines the precedence, indicating the order the script is executed. The order scripts are executed determine the placement in the GRUB boot menu. `nn` should be greater than `06` to ensure necessary scripts are executed first.

See [#Boot menu entry examples](#) for custom menu entry examples.

```
[root@archiso ~]# grub-mkconfig -o /boot/grub/grub.cfg
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-linux
Found initrd image: /boot/initramfs-linux.img
Found fallback initrd image(s) in /boot: initramfs-linux-fallback.img
Warning: os-prober will not be executed to detect other bootable partitions.
Systems on them will not be added to the GRUB boot configuration.
Check GRUB_DISABLE_OS_PROBER documentation entry.
Adding boot menu entry for UEFI Firmware Settings ...
done
```

En UEFI

```
30 grub-install --target=
31 pacman -S efibootmgr
32 grub-install --target=
```

Une fois ces commandes effectuées : on reboot

Une fois le reboot effectué, on regarde si notre configuration fonctionne :

```
ArchBetterThenWindows login: root
Password:
[root@ArchBetterThenWindows ~]# ls
[root@ArchBetterThenWindows ~]#
```

Rajouter internet : `pacstrap -K /mnt networkmanager` (AVANT LE CHMOD !)

```
Skipped: running in chroot.
root@archiso ~ # pacstrap -K /mnt networkmanager
```

2.2.1 Listing network interfaces

Both wired and wireless interface names can be found via `ls /sys/class/net` or `ip link`. Note that `lo` is the [virtual loopback interface](#) and not used in making network connections.

Wireless device names can also be retrieved using `iw dev`. See also [/Wireless#Get the name of the interface](#).

If your network interface is not listed, make sure your device driver was loaded successfully. See [/Ethernet#Device driver](#) or [/Wireless#Device driver](#).

Pour voir le statut des interfaces réseau : `ls /sys/class/net`

```
[root@ArchBetterThenWindows ~]# ls /sys/class/net
eth0 lo
```

`Ip link show dev "name interface"` -> statut des interfaces réseau

```
[root@ArchBetterThenWindows ~]# ip link show dev eth0
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 00:15:5d:19:3b:03 brd ff:ff:ff:ff:ff:ff
```

Je dois travailler avec mon `eth0`

Vérification des devices réseau sur mon windows :

☐ Obtenir une adresse IP automatiquement

☒ Utiliser l'adresse IP suivante :

Adresse IP :	<input type="text" value="172.27.32.1"/>
Masque de sous-réseau :	<input type="text" value="255.255.240.0"/>
Passerelle par défaut :	<input type="text" value="."/>

Je passe la commande : `ip address add 172.27.23.50/24 dev eth0`

```
[root@ArchBetterThenWindows ~]# ip address add 172.27.23.50/24 dev eth0
[root@ArchBetterThenWindows ~]# ip address show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
```

```
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 00:15:5d:19:3b:03 brd ff:ff:ff:ff:ff:ff
    inet 172.27.23.50/24 scope global eth0
        valid_lft forever preferred_lft forever
root@ArchBetterThenWindows ~]#
```