

Installer et utiliser accuchek

Assez de théorie, place à la pratique. Voici comment installer accuchek sur sa machine et récupérer ses mesures. Tout se passe dans un terminal, et il n'y a besoin des droits administrateur que pour l'installation.

Les prérequis

accuchek a besoin d'un compilateur C++ (C++17), de `make` et des en-têtes de **libusb**. Selon la distribution :

Distribution	Paquets
Debian, Ubuntu	<code>g++ make libusb-1.0-0-dev</code>
Fedora	<code>gcc-c++ make libusb1-devel</code>
Arch	<code>gcc make libusb</code>
Mageia	<code>gcc-c++ make lib64usb1.0-devel</code>

Par exemple, sous Debian ou Ubuntu :

```
sudo apt update && sudo apt install g++ make libusb-1.0-0-dev
```

Compiler et installer

On récupère le code, on compile, on installe :

```
git clone https://github.com/Ronnarrdd/accuchek.git
cd accuchek
make
sudo make install
```

`make install` copie deux fichiers :

- le programme dans `/usr/local/bin/accuchek` ;
- la règle udev dans `/etc/udev/rules.d/70-accuchek.rules`.

Puis on demande à udev de prendre la règle en compte :

```
sudo udevadm control --reload
sudo udevadm trigger --subsystem-match=usb
```

Pour les curieux et les empaqueteurs : `make install` respecte les variables `PREFIX`, `DESTDIR` et `UDEVDIR` (par exemple `UDEVDIR=/usr/lib/udev/rules.d`). Par défaut, le binaire est compilé de façon portable, et `make OPTFLAGS="-O3 -march=native"` l'optimise pour la machine qui compile.

Première lecture

On branche le lecteur en USB. Son écran affiche "transfert de données". On lance alors :

```
accuchek > mesures.json
```

Selon le nombre de mesures en mémoire, ça prend quelques secondes. Si la commande ne dit rien et que `echo $?` affiche `0`, c'est gagné : tout est dans `mesures.json`.

Les commandes utiles

<code>accuchek > mesures.json</code>	<code># le premier lecteur connu trouvé</code>
<code>accuchek 1 > mesures.json</code>	<code># le deuxième, si plusieurs sont branchés</code>
<code>accuchek --set-time > mesures.json</code>	<code># remet aussi l'horloge du lecteur à l'heure</code>
<code>si besoin</code>	
<code>accuchek --capture session.trace > mesures.json</code>	<code># enregistre aussi l'échange USB</code>
<code>accuchek --replay session.trace > mesures.json</code>	<code># rejoue un enregistrement, sans lecteur</code>
<code>accuchek --known-devices</code>	<code># liste des lecteurs acceptés</code>
<code>accuchek --config mes-lecteurs.txt</code>	<code># ajouter ou désactiver des modèles</code>
<code>accuchek --help</code>	

Ce qu'on obtient

Un seul objet JSON, toujours avec la même forme :

```
{
  "format": 2,
  "meter": {"manufacturer": "Roche", "model": "925", "serial": "...", "firmware": "v1.9.6",
"hardware": "G", "software": "", "system_id": "..."},
  "clock": {"meter": "2026/10/01 21:06:58", "pc": "2026/10/01 20:42:52", "offset_s": 1446,
```

```
"settable": true, "pc_synchronized": true, "action": "set"},
  "glucose": {"announced": 638, "received": 638},
  "meal": {"announced": 576, "received": 576, "unmatched": 0},
  "readings": [
    {"id": 0, "epoch": 1617009120, "timestamp": "2021/03/29 11:12", "mg/dL": 133, "mmol/L":
7.388889, "status": 0, "meal": "before_meal"}
  ]
}
```

- `meter` : la carte d'identité du lecteur ;
- `clock` : son horloge, l'écart avec le PC, et ce qui a été fait (`set`, `within_tolerance`, `not_settable`, `pc_not_synchronized`...);
- `glucose` et `meal` : ce que le lecteur annonce et ce qu'on a reçu. Si les deux nombres diffèrent, il y a un souci ;
- `readings` : les mesures. `timestamp` est l'heure affichée par le lecteur, `epoch` la même heure convertie dans le fuseau horaire du PC, heure d'été comprise.

Jouer avec les données grâce à jq

Le JSON se prête très bien à `jq`, l'outil en ligne de commande pour manipuler du JSON. Quelques exemples :

```
# la dernière mesure
jq '.readings[-1]' mesures.json

# uniquement les mesures à jeun
jq '.readings[] | select(.meal == "fasting") | {timestamp, "mg/dL"}' mesures.json

# les mesures hors échelle (HI ou LO)
jq '.readings[] | select(.range)' mesures.json

# un CSV date;valeur, à ouvrir dans un tableur
jq -r '.readings[] | select(.epoch) | [.timestamp, ."mg/dL"] | join(";")' mesures.json >
mesures.csv
```

Quand ça ne marche pas

Code de sortie	Que faire
----------------	-----------

2	Le lecteur n'est pas vu. Vérifier le câble (certains câbles micro-USB ne font que la charge), et regarder si <code>lsusb</code> affiche bien <code>173a:21d5</code> .
3	Accès refusé : installer la règle udev (voir plus haut), puis débrancher et rebrancher le lecteur.
4 ou 5	Débrancher le lecteur, attendre que son écran s'éteigne, le rebrancher et attendre "transfert de données" avant de relancer.

Pour aller plus loin, le mode debug affiche des logs et l'hexdump de chaque paquet sur la sortie d'erreur, sans jamais toucher à la sortie JSON :

```
ACCUCHEK_DBG=1 accuchek > mesures.json 2> debug.log
```

Un fichier `mesures.json`, une trace `.trace` ou un `debug.log` contiennent des données de santé. On les garde pour soi, on ne les colle pas dans un ticket ou un forum sans avoir retiré les mesures.