

Manipulations Partie 1 : Créations de mon propre module Linux

Je commence par **créer** un fichier source du module en faisant la commande :
`nano hello.c`

```
arthur@ubuntu:~$ nano hello.c
```

Ce fichier contiendra le **code du module**.

J'écris comme exemple un code qui déclare **deux fonctions** : `hello_init` (qui sera exécutée lors du **chargement** du module) et `hello_exit` (qui lui sera exécutée lors du **retrait** du module).



```
arthur@ubuntu: ~  
GNU nano 7.2 hello.c *  
#include <linux/init.h>  
#include <linux/module.h>  
  
MODULE_LICENSE("GPL");  
MODULE_AUTHOR("Arthur");  
MODULE_DESCRIPTION("Hello World");  
  
static int __init hello_init(void) {  
    printk(KERN_INFO "Hello, World!\n");  
    return 0;  
}  
  
static void __exit hello_exit(void) {  
    printk(KERN_INFO "Goodbye, World!\n");  
}  
  
module_init(hello_init);  
module_exit(hello_exit);
```

Je créer un fichier **Makefile** qui permettra de **compiler** le module en faisant :
`nano Makefile`

```
arthur@ubuntu:~$ nano Makefile
```

```
GNU nano 7.2 Makefile
obj-m += hello.o
all:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

Ensuite pour **compiler** le module j'utilise la commande : `make`, ça **générera** un fichier `hello.ko` qui est le **module compilé**.

```
arthur@ubuntu:~$ make
make -C /lib/modules/6.2.0-39-generic/build M=/home/arthur modules
make[1] : on entre dans le répertoire « /usr/src/linux-headers-6.2.0-39-generic »
warning: the compiler differs from the one used to build the kernel
The kernel was built by: aarch64-linux-gnu-gcc-12 (Ubuntu 12.3.0-1ubuntu1~23.04) 12.3.0
You are using: gcc-12 (Ubuntu 12.3.0-1ubuntu1~23.04) 12.3.0
CC [M] /home/arthur/hello.o
MODPOST /home/arthur/Module.symvers
CC [M] /home/arthur/hello.mod.o
LD [M] /home/arthur/hello.ko
BTF [M] /home/arthur/hello.ko
Skipping BTF generation for /home/arthur/hello.ko due to unavailability of vmlinux
make[1] : on quitte le répertoire « /usr/src/linux-headers-6.2.0-39-generic »
```

Pour **charger** le module j'utilise la commande : `insmod`

```
arthur@ubuntu:~$ sudo insmod hello.ko
[sudo] Mot de passe de arthur :
arthur@ubuntu:~$
```

Cette commande **charge** le **module** dans le noyau.

Pour vérifier que le module est **bien chargé** je fait la commande : `lsmod | grep hello`
Cette commande affiche **tous** les **modules chargés** et **filtre** le résultat pour trouver le **module hello**.

```
arthur@ubuntu:~$ lsmod | grep hello
hello                16384  0
arthur@ubuntu:~$
```

J'utilise `modinfo` pour voir les **informations du module**

```
arthur@ubuntu:~$ modinfo hello.ko
filename:           /home/arthur/hello.ko
description:        Hello World
author:             Arthur
license:            GPL
srcversion:         18DB59A826FB2A58A309508
depends:
name:               hello
vermagic:           6.2.0-39-generic SMP preempt mod_unload modversions aarch64
arthur@ubuntu:~$
```

Cette commande me permet **d'afficher** les différentes **informations** comme **l'auteur**, la **license** et la **description**.

Ensuite pour **retirer** le module j'utilise `rmmod`.

```
arthur@ubuntu:~$ sudo rmmod hello
arthur@ubuntu:~$
```

Cela va **décharger** le module du noyau.

Manipulations partie 2 : Module v4l2loopback

Je commence dans un premier temps pour **installer** `module-assistant`

```
arthur@ubuntu:~$ sudo apt install module-assistant
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
```

Ensuite je **prépare** le système pour **compiler** des modules avec la commande :
`sudo module-assistant prepare`

```
arthur@ubuntu:~$ sudo module-assistant prepare
Récupération des sources du noyau de la version : 6.2.0-39-generic
En-têtes du noyau disponibles dans /usr/src/linux-headers-6.2.0-39-generic
Création du lien symbolique...
apt-get install build-essential
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
build-essential est déjà la version la plus récente (12.9ubuntu3).
build-essential passé en « installé manuellement ».
Les paquets suivants ont été installés automatiquement et ne sont plus nécessaires :
  gnome-software-common libmalcontent-0-0
Veuillez utiliser « sudo apt autoremove » pour les supprimer.
0 mis à jour, 0 nouvellement installés, 0 à enlever et 41 non mis à jour.

Fait.
```

Puis je **compile** et **installe automatiquement** le module avec la commande :
`sudo module-assistant auto-install v4l2loopback`

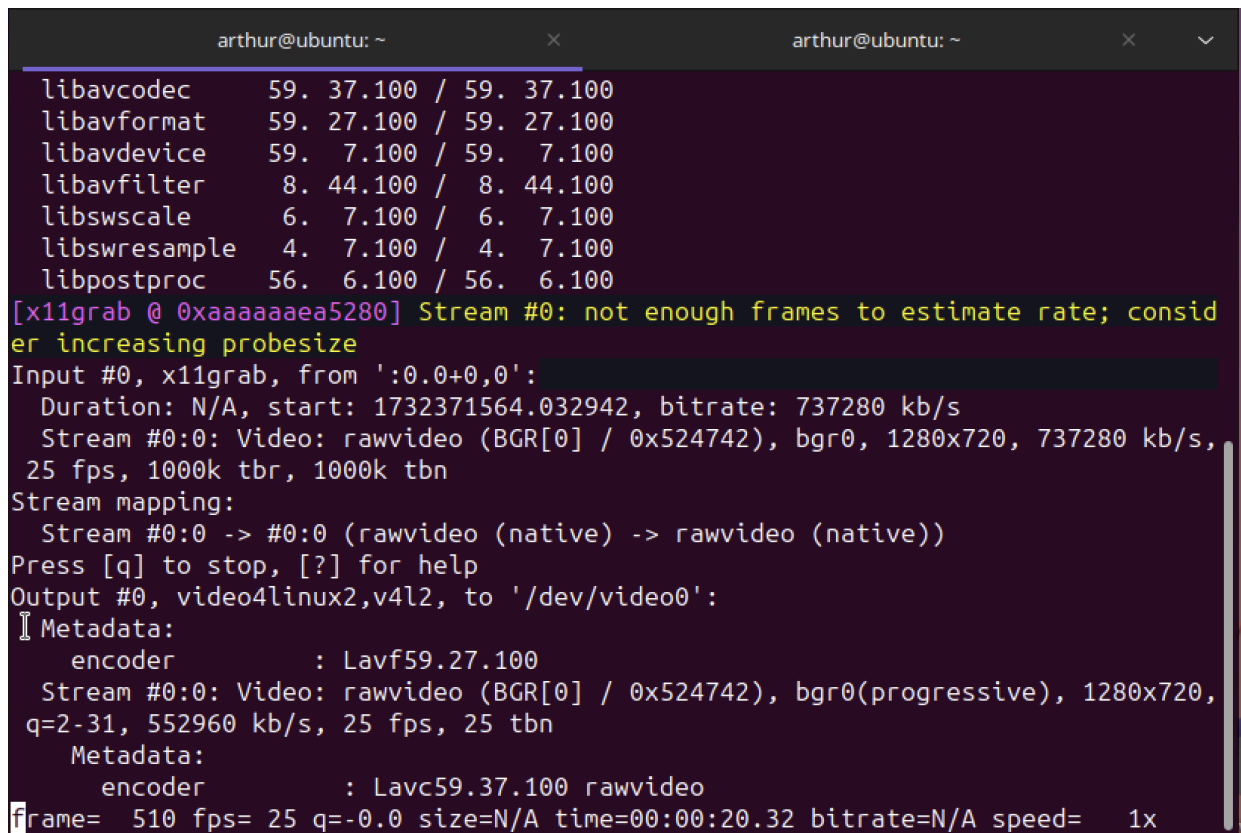
```
arthur@ubuntu:~$ sudo module-assistant auto-install v4l2loopback

Données de 1 paquets mises à jour
Récupération des sources du noyau de la version : 6.2.0-39-generic
En-têtes du noyau disponibles dans /usr/src/linux
Création du lien symbolique...
Impossible de créer le lien symbolique /usr/src/linux.
apt-get install build-essential
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
build-essential est déjà la version la plus récente (12.9ubuntu3).
Les paquets suivants ont été installés automatiquement et ne sont plus nécessaires :
  gnome-software-common libmalcontent-0-0
```

J'utilise ensuite **modprobe** pour **charger le module**

```
arthur@ubuntu:~$ sudo modprobe v4l2loopback
arthur@ubuntu:~$
```

Je continue en **testant** le **module** avec ffmpeg et ffplay, dans une **première fenêtre** de terminal j'écris la commande : `ffmpeg -f x11grab -framerate 25 -video_size 1280x720 -i :0.0+0,0 -f v4l2 /dev/video0`

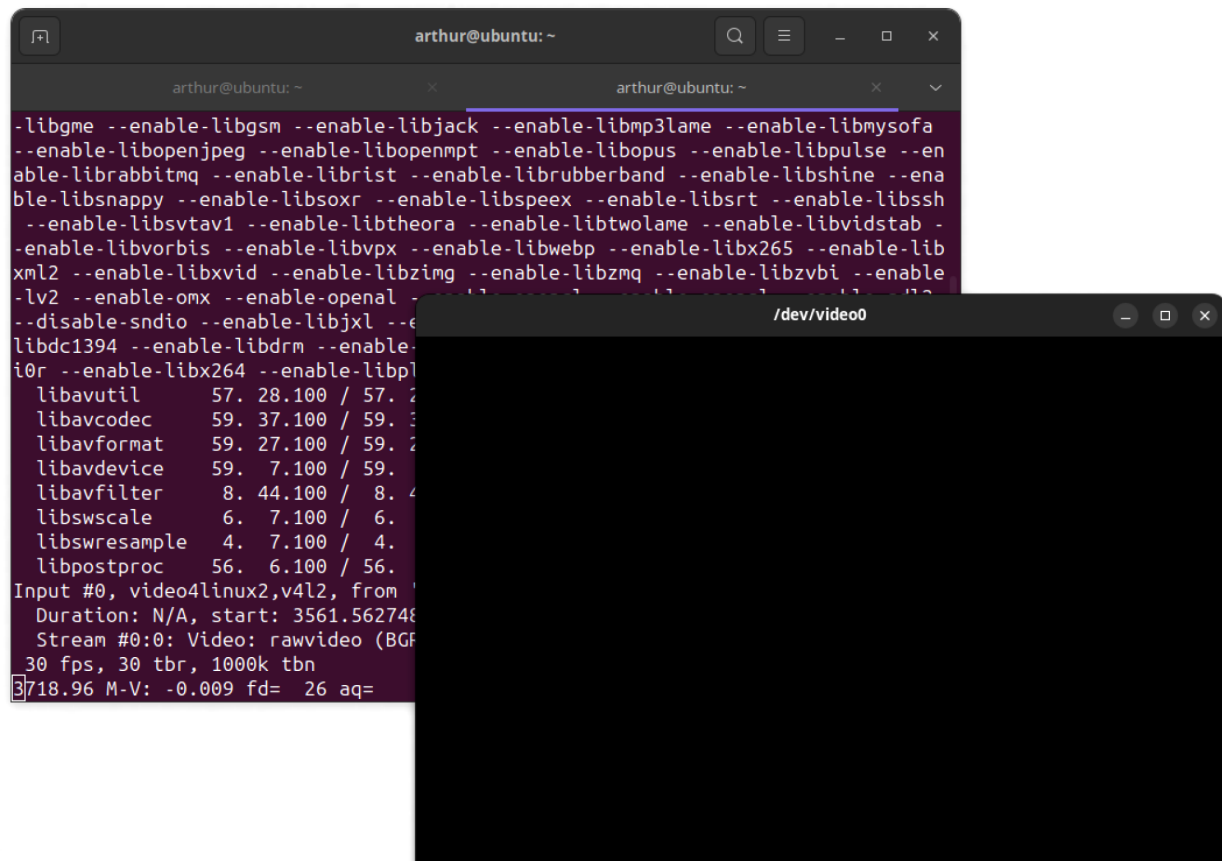


```
arthur@ubuntu: ~
libavcodec      59. 37.100 / 59. 37.100
libavformat     59. 27.100 / 59. 27.100
libavdevice     59.  7.100 / 59.  7.100
libavfilter      8. 44.100 /  8. 44.100
libswscale      6.  7.100 /  6.  7.100
libswresample   4.  7.100 /  4.  7.100
libpostproc    56.  6.100 / 56.  6.100
[x11grab @ 0xaaaaaaaa5280] Stream #0: not enough frames to estimate rate; consider increasing probesize
Input #0, x11grab, from ':0.0+0,0':
  Duration: N/A, start: 1732371564.032942, bitrate: 737280 kb/s
  Stream #0:0: Video: rawvideo (BGR[0] / 0x524742), bgr0, 1280x720, 737280 kb/s, 25 fps, 1000k tbr, 1000k tbn
Stream mapping:
  Stream #0:0 -> #0:0 (rawvideo (native) -> rawvideo (native))
Press [q] to stop, [?] for help
Output #0, video4linux2,v4l2, to '/dev/video0':
  Metadata:
    encoder      : Lavf59.27.100
  Stream #0:0: Video: rawvideo (BGR[0] / 0x524742), bgr0(progressive), 1280x720, q=2-31, 552960 kb/s, 25 fps, 25 tbn
  Metadata:
    encoder      : Lavc59.37.100 rawvideo
frame= 510 fps= 25 q=-0.0 size=N/A time=00:00:20.32 bitrate=N/A speed= 1x
```

Cette commande permet de **capturer l'écran** et le **diffuser** sur le **périphérique** `/dev/video0`

Dans une **seconde fenêtre** de terminal j'écris la commande :

```
ffplay -f v4l2 /dev/video0
```



Cette commande permet de **lire le flux diffusé** par **ffmpeg**.

Il se passe que **ffplay** va **afficher l'écran** que j'ai **capturé** avec **ffmpeg** comme s'il s'agissait d'une **webcam**.

Ainsi v4l2loopback sert à la **création de périphériques de webcam virtuels**. Cela peut-être utile pour les **visio**, les **diffusions en direct**, ou tester les **applications** utilisant des **webcams** sans avoir besoin de matériel réel.

Arthur **CHESSÉ**
B2 informatique