Question 1)

insmod helloworld.ko

-----

lsmod

Module Size Used by

helloworld 16384 0

rfcomm 65536 12

bnep 20480 2

coretemp 16384 0

[….]

-----

rmmod helloworld.ko

-----

dmesg

[…]

[80478.472492] Hello world!

[80632.398993] Goodbye world!

-----

Question 2

2) I)

#include <linux/module.h>

#include <linux/init.h>

#include <linux/types.h>

#include <linux/fs.h>

#include <linux/cdev.h>

struct cdev \* p\_vircdev;

dev\_t dev\_num;

int major\_number;

int minor\_number;

#define DEVICE\_NAME "my\_virtual\_device"

#define FIRST\_MINOR 0

#define NB\_MINOR\_ID 1

static int driver\_init(void)

{

int err;

err = alloc\_chrdev\_region(&dev\_num, FIRST\_MINOR, NB\_MINOR\_ID, DEVICE\_NAME);

if (err < 0) {

printk(KERN\_ALERT "%s: failed to obtain device numbers\n", DEVICE\_NAME);

return err;

}

p\_vircdev = cdev\_alloc();

p\_vircdev->owner = THIS\_MODULE;

err = cdev\_add(p\_vircdev, dev\_num, NB\_MINOR\_ID);

major\_number = MAJOR(dev\_num);

minor\_number = MINOR(dev\_num);

if (err < 0) {

printk(KERN\_ALERT "%s: unable to add cdev to kernel\n", DEVICE\_NAME);

return err;

}

printk(KERN\_ALERT "%s: module successfully loaded\n", DEVICE\_NAME);

printk(KERN\_INFO "\tuse \"mknod /dev/%s c %d %d\" for device file", DEVICE\_NAME, major\_number, minor\_number);

return 0;

}

static void driver\_cleanup(void)

{

unregister\_chrdev\_region(dev\_num, NB\_MINOR\_ID);

cdev\_del(p\_vircdev);

printk(KERN\_ALERT "Module %s successfully unloaded\n", DEVICE\_NAME);

}

module\_init(driver\_init);

module\_exit(driver\_cleanup);

2) ii)

insmod reg\_driver.ko

dmesg

[...]

[ 79.511247] my\_virtual\_device: module successfully loaded

[ 79.511250] use "mknod /dev/my\_virtual\_device c 246 0" for device file

rmmod reg\_driver

dmesg

[...]

[ 120.315371] Module my\_virtual\_device successfully unloaded

2) iii)

cat /proc/devices

Character devices:

[...]

226 drm

246 my\_virtual\_device

247 hidraw

248 aux

[...]

2) iv)

Ajout des droits a tous les utilistaeurs avec chmod:

chmod a=rwx /dev/my\_virtual\_device

ls -l /dev/my\_virtual\_device

crwxrwxrwx 1 root root 246, 0 Apr 27 15:34 /dev/my\_virtual\_device

3) ii)

Fonction d’écriture :

Dans un premier temps nous vérifions qu’il reste de la place sur le disque en comparant l’attribut size de notre virtual\_device avec la taille du disque. Si il ne reste plus de place alors on renvoie le code -ENOSPC.

Ensuite on alloue un char \* tampon de la taille de la variable count (passée en paramètre). On utilisera cette variable dans la fonction copy\_from\_user() qui va la remplir avec les données à ecrire sur le dique virtuel en provenance de l’utilisateur.

Enfin on rentre dans la section critique avec un semaphore. Dans cette section nous copions caractère par caratère le contenu du tampon dans le disque virtuel. Si l’onnarrive à la fin du disque on remet notre « tail » à 0 et la copie reprend jusqu’a ce que tout soit écrit ou q’il n’y y ait plus de place sur le disque. Une fois tous les caractères copiés on sort de la zone critique. Il ne reste plus qu’a maintenir à jour notre « tail » du buffer circulaire et et la taille du contenu du disque. Pour finir on renvoie le nombre de caracrères écrit.

Fonction de lecture :

Dans un premier temps on alloue un char \* tampon de la taille de count (passé en paramètre).

Ensuite on rentre dans une section critique grâce à un semaphore. Dedans nous copions caractère par caractère l’integralité du contenu du disque dans la variable tampon. Si l’on arrive à la fin du disque on remet le “head” du buffuer circulaire à 0 et la copie reprend. Une fois la copie terminée on sort de la zone critique.

Enfin le tampon ayant le contenu du disque virtuel est donné à la méthode copy\_to\_user() qui glissera le contenu du tampon vers la variable “bufUserData” passée en paramètre et qui sera retournée à l’utilisateur par son pointeur. Enfin on renvoie le nombre de caractère lus lors de l’opération.