Mise en oeuvre d'un périphérique USB sous Linux

Thierry Vaira <tvaira@free.fr>

Table des matières

Mise en oeuvre d'un périphérique USB sous Linux **1**
La norme NMEA 0183 . . . 1
Le port série virtuel . . . 2
Prise en charge . . . 2
Droits d'accès . . . 3
Tests . . . 4
Programmation en C sous Linux . . . 5
Conversions des coordonnées GPS . . . 7
Conversion format GPS en degré décimal dd.dddd . . . 7
Conversion degré décimal dd.dddd en Degres Minutes decimal-Seconds (D° M' S") . . . 7
Liens . . . 8

Site : tvaira.free.fr

Mise en oeuvre d'un périphérique USB sous Linux

Remarque : Il est conseillé de lire ou de s'aider de ce document avant de poursuive.

Ici, nous allons mettre en oeuvre le périphérique USB suivant : un **récepteur GPS de MTK**. Il émet des trames suivant la norme **NMEA 0183**. Il sera pris en charge par le système d'exploitation comme un **port série virtuel**.

La norme NMEA 0183

La norme **NMEA 0183** est une spécification pour la communication entre équipements marins, dont les équipements GPS. Elle est définie et contrôlée par la *National Marine Electronics Association* (NMEA), association américaine de fabricants d'appareils électroniques maritimes.

La norme 0183 utilise une simple communication série pour transmettre une "phrase" (*sentence*) à un ou plusieurs écoutants. Une trame NMEA utilise tous les caractères ASCII.

Exemple : *Waypoint Arrival Alarm*

```
$GPAAM,A,A,0.10,N,WPTNME*32
```

Il existe plus d'une trentaine de trames GPS différentes. Le type d'équipement est défini par les deux caractères qui suivent le `$`. Le type de trame est défini par les caractères suivants jusqu'à la virgule.

Par exemple :

```
$GPGGA,064036.289,4836.5375,N,00740.9373,E,1,04,3.2,200.2,M,,,,0000*0E
```

C'est une trame **GP** (pour GPS) de type **GGA**. La trame GGA est très courante car elle fait partie de celles qui sont utilisées pour connaître la position courante (longitude, latitude et altitude) du récepteur GPS (www.coordonnees-gps.fr).

- `$` : délimiteur de début de trame
- `GP` : Id du "parleur" (GPS)
- `GGA` : Type de trame (*Global Positioning System Fixed Data*)
- `064036.289` : Trame envoyée à 06h40m36,289s (heure UTC)
- `4836.5375,N` : Latitude ddmm.mmmm -> 48,608958° Nord = 48°36'32.25" Nord
- `00740.9373,E` : Longitude dddmm.mmmm -> 7,682288° Est = 7°40'56.238" Est
- `1` : Type de positionnement (le 1 est un positionnement GPS)
- `04` : Nombre de satellites utilisés pour calculer les coordonnées
- `3.2` : Précision horizontale ou HDOP (Horizontal dilution of precision)
- `200.2,M` : Altitude 200,2, en mètres
- `,,,,,0000` : D'autres informations peuvent être inscrites dans ces champs
- `*0E` : Somme de contrôle de parité, un simple XOR sur les caractères précédents
- `<CR><LF>` : délimiteur de fin de trame

*Remarque : Les trames NMEA font toutes référence à l'ellipsoïde **WGS84** comme base de son système de coordonnées.*

Chaque trame a sa syntaxe propre, mais selon le cas elles peuvent ou doivent se terminer, après le `*`, par un octet formant une somme de contrôle (*checksum*) qui permet de détecter une erreur dans la transmission.

La somme de contrôle à la fin de chaque phrase est le **OU EXCLUSIF** (`XOR`) de tous les octets de la phrase à l'exclusion du premier caractère (`$`) et jusqu'au caractère avant l'étoile (`*`). Cf. C implementation of checksum generation.

Site officiel : www.nmea.org

Le port série virtuel

Un port série virtuel est une **solution logicielle qui émule un port série standard**.

Cela permet généralement :

- d'augmenter le nombre de ports série (sans installation de matériel supplémentaire mais dans les limites des ressources disponibles)
- de partager les données en provenance d'un périphérique entre plusieurs applications
- de raccorder un périphérique série standard (RS232, ...) sur un port USB avec un adaptateur (manque ou absence de ports série physiques disponibles)

Pour établir une communication effective via un port série physique (RS-232) ou virtuel, il est nécessaire de définir le protocole utilisé : notamment, le **débit de la transmission (en bits/s)**, le codage utilisé, le découpage en trame, etc. La norme RS-232 laisse ces points libres, mais en pratique on utilise souvent des trames d'un caractère ainsi constituées :

- 1 bit de départ (START)
- **7 à 8 bit de données**
- **1 bit de parité optionnel (aucune, paire ou impaire)**
- **1 ou plusieurs bits d'arrêt (STOP)**

Prise en charge

Avant :

```
$ lsusb > lsusb-avant.txt
$ lsmod > lsmod-avant.txt
$ dmesg > dmesg-avant.txt
```

Brancher maintenant un périphérique USB.

Après :


```
$ lsusb > lsusb-apres.txt
$ lsmod > lsmod-apres.txt
$ dmesg > dmesg-apres.txt
```

Analyse :

```
$ diff lsusb-avant.txt lsusb-apres.txt
> Bus 002 Device 011: ID 0e8d:3329 MediaTek Inc.

$ diff dmesg-avant.txt dmesg-apres.txt
> [204555.128418] usb 2-1.4: new full-speed USB device number 10 using ehci-pci
> [204555.223664] usb 2-1.4: New USB device found, idVendor=0e8d, idProduct=3329
> [204555.223669] usb 2-1.4: New USB device strings: Mfr=3, Product=4, SerialNumber=0
> [204555.223672] usb 2-1.4: Product: GPS Receiver
> [204555.223674] usb 2-1.4: Manufacturer: MTK
> [204555.257427] cdc_acm 2-1.4:1.1: ttyACM0: USB ACM device
> [204555.257994] usbcore: registered new interface driver cdc_acm
> [204555.257997] cdc_acm: USB Abstract Control Model driver for USB modems and ISDN adapters

$ diff lsmod-avant.txt lsmod-apres.txt
> cdc_acm                26991  2

$ modinfo cdc_acm
filename:       /lib/modules/3.8.0-44-generic/kernel/drivers/usb/class/cdc-acm.ko
alias:          char-major-166-*
license:        GPL
description:    USB Abstract Control Model driver for USB modems and ISDN adapters
author:         ...
srcversion:     AD91845189110F8BEC40793
...
alias:          usb:v0E8Dp3329d*dc*dsc*dp*ic*isc*ip*in*
alias:          usb:v0E8Dp0003d*dc*dsc*dp*ic*isc*ip*in*
...

$ ls -l /dev/ttyACM0
crw-rw---- 1 root dialout 166, 0 sept. 18 16:24 /dev/ttyACM0
```

- le périphérique USB a été détecté physiquement par le noyau (cf. lsusb et dmesg)
- c'est un "GPS Receiver" de MTK (idVendor=0e8d, idProduct=3329)
- ce périphérique (idVendor=0e8d, idProduct=3329) peut être pris en charge par le module chargeable `cdc_acm` (cf. lsmod et modinfo)
- le pilote de périphérique (*driver*) associé à ce matériel (`cdc_acm`) a été chargé par le système (cf. dmesg et lsmod)
- le périphérique sera accessible par le fichier spécial `ttyACM0` dans `/dev` (cf.)

Attention : le périphérique ne sera accessible en lecture/écriture que par l'utilisateur propriétaire `root` et les membres du groupe `dialout`.

Droits d'accès

Par défaut, le système applique la politique des droits d'accès `rw-rw----` pour ce type de périphérique USB :

```
$ ls -l /dev/ttyACM0
crw-rw---- 1 root dialout 166, 0 sept. 18 16:24 /dev/ttyACM0
```

Ici, le périphérique sera accessible en lecture/écriture par l'utilisateur propriétaire `root` et les membres du groupe `dialout`. Tous les autres (*other*) utilisateurs auront aucun accès.

Pour modifier cette situation, vous avez les possibilités suivantes :

- changer les droits manuellement (à chaque fois !) : `$ sudo chmod 666 /dev/ttyACM0`
- ajouter l'utilisateur demandeur (ici toto) dans le groupe `dialout` : `$ sudo adduser toto dialout`

Pour une gestion automatique, il faudra passer par `udev` qui est maintenant le service qui prend en charge le répertoire `/dev`.

```
$ ls -l /dev/ttyACM0
crw-rw---- 1 root dialout 166, 0 sept. 18 16:24 /dev/ttyACM0
```



```
$ sudo vim /etc/udev/rules.d/51-ttyusb.rules
SUBSYSTEMS=="usb", ATTRS{idVendor}=="0e8d", ATTRS{idProduct}=="3329", MODE="0666"

$ ls -l /dev/ | grep ACM
crw-rw-rw-  1 root dialout   166,   0 sept. 18 17:08 ttyACM0
```

Il est aussi possible de créer un lien symbolique au choix :

```
$ sudo vim /etc/udev/rules.d/51-ttyusb.rules
SUBSYSTEMS=="usb", ATTRS{idVendor}=="0e8d", ATTRS{idProduct}=="3329", MODE="0666", SYMLINK+="gps"

$ ls -l /dev/ | grep "\(ACM\|gps\)"
lrwxrwxrwx  1 root root             7 sept. 18 17:10 gps -> ttyACM0
crw-rw-rw-  1 root dialout   166,   0 sept. 18 17:10 ttyACM0
```

Ou de fixer le nom du fichier spécial :

```
$ sudo vim /etc/udev/rules.d/51-ttyusb.rules
SUBSYSTEMS=="usb", ATTRS{idVendor}=="0e8d", ATTRS{idProduct}=="3329", MODE="0666", NAME="ttyUSB1",
SYMLINK+="gps"

$ ls -l /dev/ | grep "\(gps\|USB\)"
lrwxrwxrwx  1 root root             7 sept. 18 17:12 gps -> ttyUSB1
crw-rw-rw-  1 root dialout   166,   0 sept. 18 17:12 ttyUSB1
```

Tests

On peut tester la réception de phrases NMEA 0183 avec la commande `screen` (`Ctrl-a k` pour sortir) ou avec `picocom` (`Ctrl-a Ctrl-x` pour sortir) :

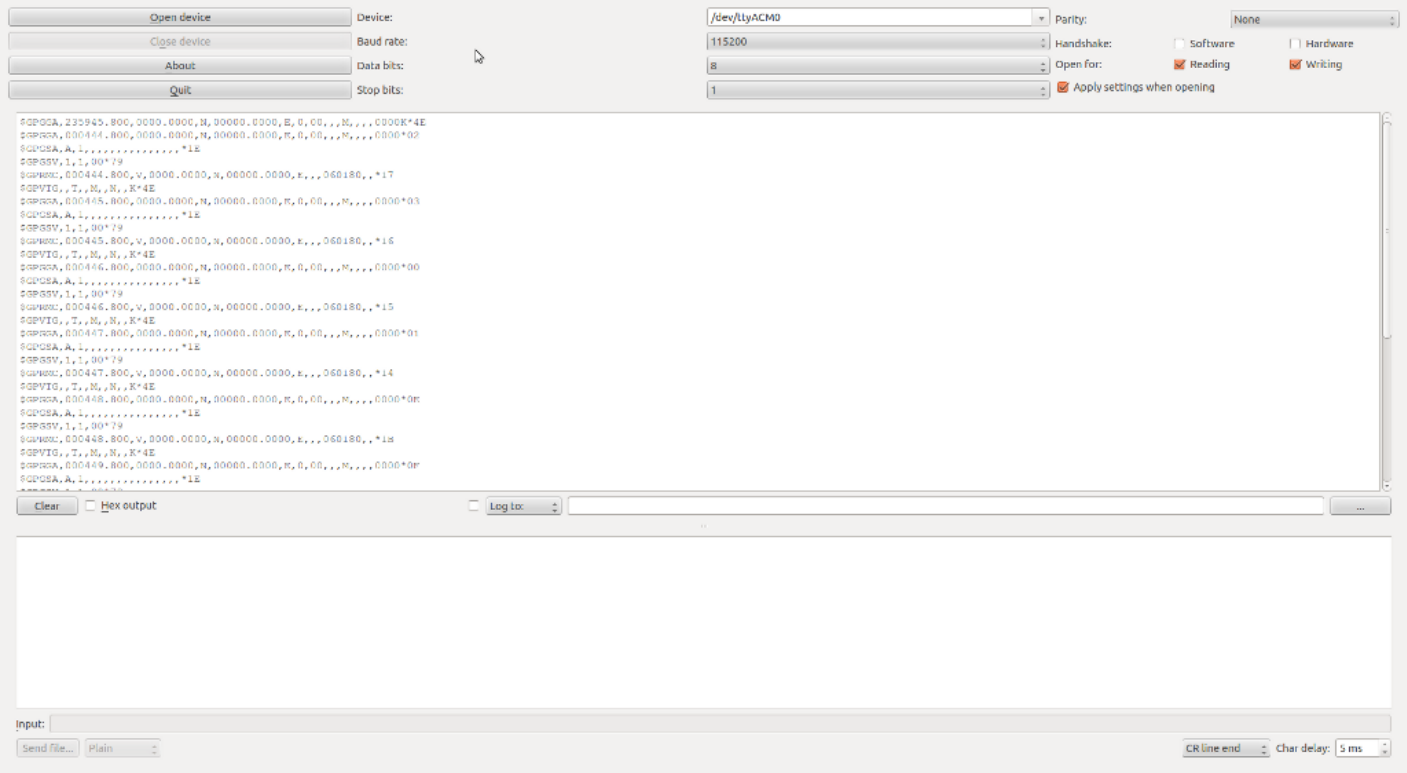
```
$ screen /dev/ttyACM0 115200
$GPRMC,000936.799,V,0000.0000,N,00000.0000,E,,,060180,,*10
$GPVTG,,T,,M,,N,,K*4E
$GPGGA,000937.799,0000.0000,N,00000.0000,E,0,00,,,M,,,,0000*04
$GPGSA,A,1,,,,,,,,,,,,,,,*1E
$GPGSV,1,1,00*79
$GPRMC,000937.799,V,0000.0000,N,00000.0000,E,,,060180,,*11
$GPVTG,,T,,M,,N,,K*4E
$GPGGA,000938.799,0000.0000,N,00000.0000,E,0,00,,,M,,,,0000*0B
...

$ picocom -b 115200 /dev/ttyACM0
picocom v1.4

port is        : /dev/ttyACM0
flowcontrol    : none
baudrate is    : 115200
parity is      : none
databits are   : 8
escape is      : C-a
noinit is      : no
noreset is     : no
nolock is      : no
send_cmd is    : ascii_xfr -s -v -l10
receive_cmd is : rz -vv

Terminal ready
$GPGGA,064509.842,4405.2015,N,00457.8701,E,0,00,,,M,,,,0000*05
$GPGSA,A,1,,,,,,,,,,,,,,,*1E
$GPGSV,3,1,12,29,80,028,,31,61,278,,25,55,092,,21,29,178,*73
$GPGSV,3,2,12,26,28,296,,12,17,102,,02,14,041,,14,06,218,*74
$GPGSV,3,3,12,05,06,076,,20,03,123,,23,02,333,,16,02,290,*77
$GPRMC,064509.842,V,4405.2015,N,00457.8701,E,,,041015,,*1E
$GPVTG,,T,,M,,N,,K*4E
```



```
...
```

Remarque : 115200 est le débit de la "ligne" en bits/s.

Ou avec le logiciel `cutecom` :



Programmation en C sous Linux

```c
// Lit une trame NMEA 183 sur le port série virtuel

// Attention : vérifiez le nom du fichier de périphérique et la configuration du port série

/*************************************************************
*   gcc -O2 -o nom_executable nom_source
*************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <termios.h>
#include <unistd.h>
#include <sys/fcntl.h>

// On fixe la taille maximum à 128 octets
#define MAX 128

//#define DEBUG

int main(void)
{
    int fd; // descripteur de fichier
    char nomPeripherique[16] = "/dev/ttyACM0"; // nom du fichier special
    struct termios termios_p; // parametres du port
    char* buffer = (char *)NULL; // un buffer
    char c = 0; // un caractère
    int fini = 0; // fin de reception d'une trame
```



```
int i = 0, n = 0;
int debutTrame = 0; // debut de reception d'une trame

/* Ouverture de la liaison serie */
fd = open(nomPeripherique, O_RDWR | O_NOCTTY | O_NONBLOCK);
if ( fd == -1 )
{
    perror("open"); // message d'erreur système
    //printf("Erreur d'ouverture du peripherique !\n"); // message d'erreur client
    exit(1);
}
fflush(NULL);
  //printf("Peripherique ouvert avec succes !\n");

/* Lecture des parametres courants  */
tcgetattr(fd, &termios_p);
/* On ignore les BREAK et les caracteres avec erreurs de parite */
termios_p.c_iflag = IGNBRK | IGNPAR;
/* Pas de mode de sortie particulier */
termios_p.c_oflag = 0;
/* Liaison a 9600 bps, 8 bits de donnees, pas de parite, lecture possible */
termios_p.c_cflag = B9600 | CS8 | CREAD | CLOCAL;
termios_p.c_cflag &= ~PARODD ;
termios_p.c_cflag &= ~PARENB;
/* Desactive le mode canonique */
termios_p.c_lflag = 0;
/* Caracteres immediatement disponibles */
termios_p.c_cc[VMIN] = 1;
termios_p.c_cc[VTIME] = 0;

/* Sauvegarde des nouveaux parametres */
if(tcsetattr(fd,TCSANOW,&termios_p)== -1)
{
  perror("tcsetattr");
  close(fd);
  exit(1);
}
//printf("Sauvegarde de la nouvelle configuration avec succes !\n");

/* Lecture de MAX octets */
buffer = (char *)malloc((MAX+1) * sizeof(char)); // on alloue un buffer
i = 0;
debutTrame = 0;
do
{
    n = read(fd, &c, 1);
    #ifdef DEBUG
    printf("read : %d -> 0x%02X ", n, (unsigned char)c);
    #endif
    // est-ce qu'on a lu 1 caractère ?
    if(n == 1)
    {
        // est-ce le début d'une trame ?
        if(c == '$')
        {
          debutTrame = 1;
        }
        // est-ce qu'on est entrain de lire une trame ?
        if(debutTrame == 1)
        {
          // on copie le caractère lu dans le buffer
          *(buffer + i) = c;
          i++;
        }
        // est-ce que c'est la fin de la trame qu'on est entrain de lire ?
        if (debutTrame == 1 && c == '\n')
```



```
            {
              fini = 1;
              debutTrame = 0;
            }
        }
        /*else
        {
            if(n == -1)
            {
                perror("read"); // message erreur système
                //printf("Erreur de lecture\n"); // message client
                //fini = 1;
            }
            else
            {
                printf("Aucune lecture !\n"); // message warning
            }
        }*/
    }
    while(i <= MAX && fini == 0);

    *(buffer + i) = 0x00; // fin de chaîne (attention à i > MAX !)

    printf("Trame lue :\n%s\n", buffer);

    /* Fermeture du port série */
    close (fd);
    //printf("Fermeture du peripherique avec succes !\n");

    free(buffer); // on libère la mémoire allouée

    return 0;
}
```

```
$ gcc -O2 test-port-com-usb.c

$ ./a.out
Trame lue :
$GPZDA,001448.799,06,01,1980,,.0000,E,0,00,0,00,,,M,,,,0000*01
```

Code source : test-port-com-usb-linux.c

Conversions des coordonnées GPS

"$GPGGA,084222.000,4405.2015,N,00457.8908,E,1,05,1.7,26.5,M,,,,0000*3F"
Latitude et longitude dans le format : ddmm.mmmm et dddmm.mmmm

Conversion format GPS en degré décimal dd.dddd

dd + mm.mmmm/60
4405.2015 —> 44 + 05.2015/60 = 44,0867 degres

Conversion degré décimal dd.dddd en Degres Minutes decimal-Seconds (D° M' S")

d = 44.0823
D = int(d) —> 44
M = int((d - D) x 60) —> ((44.0823 - 44) x 60) = 4,938 = 4
s = (d - D - M/60) x 3600 —> (44.0823 - 44 - 4/60) x 3600
ou s = (d - D) x 3600 - M x 60
44.0867 —> 44° 5' 12.12"

Liens

- www.coordonnees-gps.fr
- www.sunearthtools.com

Retour au sommaire

