EISE OSK

Programmation avec le module noyau `ppdev`

Utilisation de ppdev

Le module noyau **`ppdev`** fait partie du système Linux standard. C'est un gestionnaire de périphérique de type caractère (char device driver) qui permet de piloter individuellement les broches du port parallèle depuis une application utilisateur. Avec ce module, il n'est plus nécessaire d'être **`root`** pour commander l'écran lcd et le thermomètre.

Avec **`ppdev`**, l'accès au port parallèle se fait de manière canonique, comme avec tout gestionnaire de périphérique. En particulier, le port parallèle est représenté comme un fichier dont le nom est **`/dev/parport0`**. Pour le manipuler, la méthode standard consiste à ouvrir ce fichier avec l'appel système **`open()`**, puis à écrire dedans au moyen de **`write()`** ou de lire des données avec **`read()`**. Pour configurer le périphérique dans un certain mode, il faut utiliser l'appel système **`ioctl()`**. Pour notre application avec **`ppdev`**, seul l'appel système **`ioctl()`** va en fait être utilisé.

Remarques concernant l'utilisation du module noyau ppdev

Pour pouvoir profiter de ce module noyau, il faut vérifier qu'il est bien installé sur votre système. Pour cela, il suffit de lancer la commander **`/sbin/lsmod`**, qui permet de lister les modules noyau actuellement opérationnels et d'y repérer le module **`ppdev`**. Vous pouvez aussi vérifier que le fichier représentant le périphérique, **`/dev/parport0`**, existe bien.

Pour programmer une application utilisant les appels systèmes précités et accéder au port parallèle, vous devez inclure les fichiers suivants.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <math.h>
#include <sys/io.h>

#include <fcntl.h>
#include <errno.h>
#include <linux/ppdev.h>
#include <linux/parport.h>
#include <sys/ioctl.h>
#include <sys/stat.h>
#include <sys/types.h>
```

Description des appels système :

```
int open(char *pathname, int mode)
```

Cet appel système ouvre le fichier de chemin d'accès **pathname** dans le mode **mode**. Pour accéder en écriture et en lecture dans ce fichier, mode doit être égal à **O_RDWR**. L'appel système retourne le numéro de descripteur de fichier ouvert ou -1 s'il y a eu une erreur

int close(int filedesc)

Cet appel système permet de fermer le fichier de descripteur **filedesc**. A utiliser lorsque le programme se termine.

int ioctl(int filedesc, int cmd, …)

Cet appel système est celui par lequel toutes vos actions vont s'effectuer. Il peut prendre de multiples formes :

> **int ioctl(fd, PPCLAIM)** doit être appelé dès que le fichier est ouvert, pour réclamer l'usage exclusif du périphérique port parallèle. Retourne -1 si erreur.
>
> **int ioctl(fd, PPRELEASE)** doit être appelé avant que le fichier soit fermé, pour libérer l'usage exclusif du périphérique. Retourne -1 si erreur.
>
> **int ioctl(fd, numreg, char *byte)** est la fonction que vous allez principalement utiliser. **numreg** peut prendre les valeurs **PPRDATA** (pour lire le port data), **PPRSTATUS** (pour lire le port d'état), **PPRCONTROL** (pour lire le port de contrôle), **PPWDATA** (pour écrire sur le port data), **PPWCONTROL** (pour écrire sur le port contrôle). Pour une lecture, ***byte** contient après appel la valeur lue. Pour une écriture, il faut affecter ***byte** avant appel. Retourne -1 si erreur.

Ainsi, pour écrire la valeur **0xaa** sur le port data (=0x378) du port parallèle, il faut écrire :

```
char byte=0xaa ;
if (ioctl(fd,PPWDATA,&byte))
{
     printf("erreur");
     exit(1);
}
```

Pour lire la valeur du registre d'état (=0x379) :

```
if (ioctl(fd,PPRSTATUS,&byte))
{
     printf("erreur");
     exit(1);
}
// byte contient maintenant la valeur du registre d'état
```

Questions

1) Ecrire la fonction qui permet d'ouvrir correctement le périphérique. Le descripteur de fichier est déclaré en variable globale. Ne pas oublier de réclamer après l'usage exclusif de l'interface.

2) Ecrire la fonction qui permet de fermer le fichier. Ne pas oublier de libérer avant l'usage exclusif de l'interface.

3) Réécrire l'équivalent utilisateur des fonctions **`outb()`** et **`inb()`** étudiées dans le précédent TP.

4) Modifier les fonctions d'accès **`LCD_E_HIGH()`**, **`LCD_E_LOW()`** en conséquence.

5) Faire fonctionner l'ensemble de l'application, thermomètre inclus.