1. Mémoire virtuelle et mapping mémoire

La carte Raspberry Pi est composée d'un système sur puce (System-on-Chip, SoC) BCM2835 de chez Broadcom. Ce SoC est composé de deux Memory Management Units (MMUs).

La première MMU est intégrée au processeur ARM 11 du SoC, elle traduit les adresses virtuelle des processus en adresses physiques. Cette première MMU est paginée. La seconde est une MMU segmentée "gros grain" (au niveau du SoC lui-même). La première MMU est une unité de gestion des translations d'adresse "classique" tandis que la seconde MMU permet de remanier le placement mémoire des périphériques du SoC.

1. Qu'est-ce que la mémoire physique ? (réponse)
2. Qu'est-ce que la mémoire virtuelle ? (réponse)
3. Qu'est-ce qu'un processus ? (réponse)
4. Quelle est l'intérêt de la mémoire virtuelle ? (réponse)
5. Qu'est-ce qu'une MMU ? (réponse)
6. Qu'est-ce qu'un défaut de page ? Est-ce grave ? (réponse)

2. Accès aux registres mappés en mémoire

Nous fonctionnons sur un système qui utilise un mécanisme de mémoire virtuelle et dès lors l'accès à certaines zones mémoires est protégé par le système d'exploitation, ici Linux. C'est le cas notamment pour les zones mémoires correspondant aux contrôleurs d'entrées/sorties. Néanmoins, un peut quand même accéder à ces zones de mémoires grâce à l'appel système mmap, dont le prototype est le suivant:

void * mmap (void * addr, size_t length, int prot, int flags, int fd, off_t offset);

et son opposé:

int munmap (void * addr, size_t length);

L'appel système mmap permet de récupérer un pointeur vers une zone mémoire donnée en paramètres. À l'inverse, munmap permet de libérer l'association entre le pointeur fourni par

mmap et la zone mémoire associée lors de l'appel à mmap.

7. Quel est l'intérêt ou le besoin de protéger les zones mémoires liées aux contrôleurs d'entrées/sorties ? (réponse)

Pour accéder aux contrôleurs d'entrées/sorties mappés en mémoire, on utilise le mapping configuré dans le noyau Linux. Ce mapping est accessible depuis le fichier /dev/mem.

8. Qu'est-ce que le fichier /dev/mem ? Est-ce un vrai fichier ? (réponse)
9. Comment utilise-t-on le fichier /dev/mem ? (réponse)
10. Sachant que l'adresse de base du contrôleur de GPIO est 0x20200000, quel serait le code C qui permettrait d'accéder au registre GPFSEL0 du contrôleur de GPIO ? (réponse)

3. Réalisation d'un programme pour faire clignoter une LED

11. Quelles sont les grandes étapes pour réaliser un programme qui fasse clignoter une LED à une fréquence donnée ? (réponse)
12. Quelles sont les fonctions C associées ? (réponse)

Réponses

1. La mémoire physique est, en principe, la vraie mémoire qui est accédée par un espace d'adressage. Derrière chaque adresses physique, il y a de la vraie mémoire ou un registre de configuration de périphérique.

2. La mémoire virtuelle est un espace d'adressage (un ensemble d'adresses) dans lequel s'exécute un processus.

3. Un processus est un conteneur de ressource permettant à un programme de s'exécuter. Les ressources sont :

    - un espace d'adressage
    - un programme
    - un ensemble de fichiers ouverts
    - ...

    - un ou plusieurs fils d'exécution (threads) ayant chacun
        - un contexte de processeur
        - une pile d'exécution

4. La mémoire virtuelle a plusieurs avantages :

    - La mémoire virtuelle permet d'isoler les processus entre eux.
    - La mémoire virtuelle permet de produire du code binaire sans connaître les adresses effectivement disponibles au moment de l'exécution.
    - La mémoire virtuelle d'étendre la mémoire utilisable au delà de la mémoire physique réellement disponible (grâce au swap disque).

5. Chaque espace virtuel va être mappé sur l'espace physique, c'est-à-dire que lorsqu'un processeur exécute une instruction d'accès à la mémoire. Il commence par émettre l'adresse de l'instruction puis l'adresse de la donnée. Ces adresses sont virtuelles et sont traduites en adresses physiques par la MMU (Memory Management Unit) sur la base d'une table de page qui associe une page physique à chaque page virtuelle donnée à l'application avec des droits d'accès.

6. Un défaut de pages survient lorsque que le processeur demande l'accès à une adresse virtuelle pour laquelle l'OS n'a pas encore attribuée de page physique. Ce n'est pas forcément un problème, c'est même le cas normal, puisque l'OS n'attribue des pages physiques que si c'est nécessaire.

7. Sûreté et sécurité :

    - sûreté pour ne pas faire un mauvais usage du matériel
    - sécurité pour ne pas utiliser les droits pour volontairement casser le matériel

8. C'est un pseudo-fichier qui permet d'accéder à la mémoire physique au travers de l'abstraction des fichiers.

9. Il faut avoir les droits du root pour ouvrir ce fichier. Ce n'est évidemment pas un fichier, ce n'est qu'une abstraction.

10. il faut l'ouvrir avec open(), on peut ensuite utiliser read() et write() mais il est plus propre d'utiliser mmap() qui permet d'accéder à la mémoire comme de la mémoire (comme un tableau d'octets).

```
mmap_result = mmap (
```

```
    NULL                    // adresse virtuelle souhaitée, laissée NULL
quand on laisse choisir l'OS
    , RPI\_BLOCK\_SIZE         // nombre d'octets demandés qui doit être
un multiple de la taille d'une page
    , PROT_READ | PROT_WRITE // type d'accès autorisé par le programme
    , MAP_SHARED             // si plusieurs processus map le même zone,
ils verront la même copie (!=MAP_PRIVATE)
    , mmap_fd                // file descriptor : identifiant de /dev/mem
donné par open
    , BCM2835\_GPIO\_BASE );   // décalage par rapport au premier octet de
la mémoire
   (unsigned volatile int *) GPSEL0 = mmap_result; // volatile pour ne pas
avoir une mise en cache dans les registres
   *GPSELO = value;
```

11. Clignoter une led :

```
-   ouvrir /dev/mem
-   mapper la zone où se trouve les registres de contrôle des GPIO
-   configurer la broche GPIO choisie en sortie.
-   faire une boucle
    -   broche <= 0
    -   attendre
    -   broche <= 1
    -   attendre
```

12. les fonctions nécessaires sont:

```
-   open
-   mmap
-   sleep (ou un équivalent)
```