

TD UTC502 : Notions de base sur les systèmes d'exploitation

A Rappelez en quelques lignes le rôle des interruptions dans le mécanisme des entrées-sorties.

B Contrôleur d'interruptions matérielles

On considère une machine admettant 8 niveaux d'interruptions matérielles numérotées de 0 à 7, le niveau d'interruptions 0 étant le plus prioritaire et le niveau 7 le moins prioritaire. Le processeur dispose de deux broches, une broche INT sur laquelle lui parvient le signal d'interruption, une broche INTA avec laquelle il acquitte les interruptions. Les 8 niveaux d'interruptions sont gérés par un contrôleur d'interruptions qui possède deux registres de 8 bits chacun, lui permettant de gérer les requêtes d'interruptions et leurs priorités. Le premier registre appelé registre **ISR** mémorise les interruptions en cours de traitement en positionnant à 1 le bit correspondant aux interruptions en cours de traitement. Le second registre **IRR** mémorise les interruptions reçues par le contrôleur en positionnant à 1 le bit correspondant à l'interruption reçue, cette mémorisation durant jusqu'à ce que l'interruption ait été traitée par le processeur.

Les priorités sur les interruptions fonctionnent de la manière suivante : lorsque le processeur traite une interruption de niveau i (le bit i du registre ISR est alors à 1), toutes les interruptions de priorité inférieure ou égale à i parvenant au contrôleur sont mémorisées dans le registre IRR, mais elles ne sont pas délivrées au processeur. Elles seront délivrées selon leur ordre de priorité, ultérieurement, dès que le traitement de l'interruption i aura été achevé. Une interruption de priorité supérieure au niveau i est au contraire délivrée au processeur et interrompt le traitement de l'interruption i en cours qui est repris ultérieurement, une fois le traitement de l'interruption plus prioritaire achevé.

1. Considérez l'état courant du contrôleur et du processeur donné sur la figure. Que se passe-t-il ? Que va faire le processeur ?

2. En partant de l'état précédent, décrivez la suite des opérations tant au niveau du contrôleur (évolution des registres ISR et IRR) que du processeur si l'on considère que surviennent simultanément au contrôleur les interruptions 1 et 5.

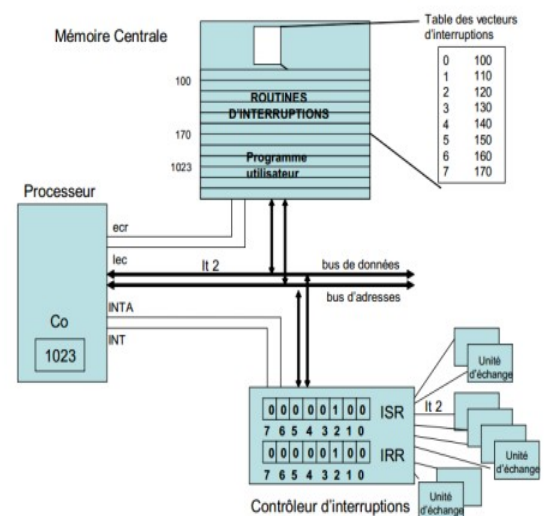


Figure 1: État courant du contrôleur et du processeur

C PRIMITIVES DU VFS

C.1 Opérations sur les fichiers

C.1.1 Ouverture d'un fichier

L'ouverture d'un fichier s'effectue par un appel à la primitive `open()` dont le prototype est donné :

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
int open(const char *ref, int mode_ouv, mode_t mode);
```

Le premier paramètre `*ref` spécifie le chemin du fichier à ouvrir dans l'arborescence du système de gestion de fichiers.

Le paramètre `mode_ouv` est une combinaison de constantes permettant de spécifier les options sur le mode d'ouverture. Ces constantes peuvent être combinées entre elles à l'aide de l'opération de OU binaire « | ». Ce sont :

- `O_RDONLY`, `O_WRONLY`, `O_RDWR` pour demander une ouverture respectivement en lecture seule, écriture seule ou lecture et écriture ;
- `O_CREAT` pour demander la création du fichier s'il n'existe pas. Cette constante peut être combinée avec la constante `O_EXCL` auquel cas la demande de création échoue si un fichier de même nom existe déjà ;
- `O_TRUNC` pour demander la suppression du contenu du fichier si celui-ci existe déjà ;
- `O_APPEND` pour demander l'ouverture du fichier en mode ajout, toutes les écritures ayant lieu à la fin du fichier ;
- `O_SYNC` pour demander la gestion du fichier en mode synchrone, ce qui implique que toute écriture réalisée sur le fichier est immédiatement recopiée sur le disque.

Le troisième paramètre de l'appel n'est positionné que si la création du fichier a été demandée, c'est-à-dire si l'option `O_CREAT` a été utilisée. Elle permet de spécifier les droits d'accès associés au fichier. Les valeurs admises sont :

- `S_ISUID`, `S_ISGID`, `S_ISVTX`, activation respective des bits setuid, setgid, sticky ;
- `S_IRUSR`, `S_IWUSR`, `S_IXUSR`, `S_IRWXU`, activation respective des droits en lecture, écriture, exécution et lecture/écriture/exécution pour le propriétaire du fichier ;
- `S_IRGRP`, `S_IWGRP`, `S_IXGRP`, `S_IRWXG`, activation respective des droits en lecture, écriture, exécution et lecture/écriture/exécution pour le groupe du propriétaire du fichier ;
- `S_IROTH`, `S_IWOTH`, `S_IXOTH`, `S_IRWXO`, activation respective des droits en lecture, écriture, exécution et lecture/écriture/exécution pour les autres.

C.1.2 Fermeture d'un fichier

La fermeture du fichier par un processus s'effectue par un appel à la primitive `close()` dont le prototype est :

```
#include <unistd.h>
int close(int desc);
```

Le paramètre `desc` correspond au descripteur de fichier obtenu lors de l'ouverture du fichier.

C.1.3 Lecture et écriture dans un fichier

Le système Linux considère les fichiers comme une suite d'octets non typés et sans aucune structure. C'est donc l'application qui par la définition de types structurés applique une structure au fichier.

La lecture dans un fichier s'effectue à l'aide de la primitive `read()` dont le prototype est :

```
#include <unistd.h>
ssize_t read(int desc, void *ptr_buf, size_t nb_octets );
```

Cette opération de lecture lit dans le fichier correspondant au descripteur `desc`, dans le buffer de réception pointé par `ptr_buf`, un nombre de `nb_octets` octets depuis l'octet où est positionné le pointeur du fichier. Le pointeur du fichier est incrémenté du nombre d'octets lus.

La primitive renvoie le nombre de caractères effectivement lus (0 si la fin de fichier est atteinte) et la valeur `-1` en cas d'échec. La variable `errno` prend l'une des valeurs suivantes :

- `EBADF`, le descripteur est invalide ;
- `EFAULT`, `buf` n'est pas une adresse valide ;
- `EINTR`, l'appel système a été interrompu par la réception d'un signal ;
- `EINVAL`, le fichier pointé par `fd` ne peut pas être lu ;
- `EIO`, une erreur d'entrées-sorties est survenue ;
- `EISDIR`, le chemin spécifié est un répertoire.

L'écriture dans un fichier s'effectue à l'aide de la primitive `write()` dont le prototype est :

```
#include <unistd.h>
ssize_t write(int desc, void *ptr_buf, size_t nb_octets );
```

Cette opération d'écriture place dans le fichier de descripteur `desc`, le contenu du buffer pointé par `ptr_buf`, un nombre d'octets correspondant à `nb_octets`.

Le pointeur du fichier est incrémenté du nombre d'octets écrits.

La primitive renvoie le nombre de caractères effectivement écrits et la valeur `-1` en cas d'échec. La variable `errno` prend l'une des valeurs suivantes :

- `EBADF`, le descripteur est invalide ;
- `EFAULT`, `buf` n'est pas une adresse valide ;
- `EINTR`, l'appel système a été interrompu par la réception d'un signal ;
- `EINVAL`, le fichier pointé par `fd` ne peut pas être lu ;
- `EIO`, une erreur d'entrées-sorties est survenue ;
- `EISPIPE`, le descripteur référence un tube sans lecteur ;

- ENOSPC, le système de fichiers est saturé ;
- EISDIR, le chemin spécifié est un répertoire.

C.1.4 Se positionner dans un fichier

Les lectures et les écritures sur un fichier s'effectuent de façon séquentielle. Il est cependant possible de modifier la position courante du pointeur de fichier à l'aide de la primitive `lseek()` dont le prototype est :

```
#include <unistd.h>
off_t lseek (int desc, off_t dep, int option);
```

Le pointeur du fichier désigné par `desc` est modifié d'une valeur égale à `dep` octets selon une base spécifiée dans le champ `option`. Ce champ `option` peut prendre trois valeurs :

- `SEEK_SET`, le positionnement est effectué par rapport au début du fichier ;
- `SEEK_CUR`, le positionnement est effectué par rapport à la position courante ;
- `SEEK_END`, le positionnement est effectué par rapport à la fin du fichier.

La primitive retourne la position courante en octets par rapport au début du fichier à la suite du positionnement, et la valeur `-1` en cas d'échec. La variable `errno` prend l'une des valeurs suivantes :

- `EBADF`, le descripteur est invalide ;
- `EINVAL`, `option` n'a pas une valeur admise ;
- `EISPIPE`, le descripteur référence un tube sans lecteur.

C.1.5 Compléter le code

```
/* *****
/* Exemple de manipulation d'un fichier: */
/* création, positionnement, fermeture */
/* *****

#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

typedef struct student {
char nom[10];
int note;
} eleve;

int main()
{

int fd, i, ret;
eleve un_eleve;
fd = open (A COMPLETER);
if (fd == -1)
perror ("prob open");
i = 0;
while (i<4)
{
printf ("Donnez le nom de l'élève \n");
scanf ("%s", un_eleve.nom);
printf ("Donnez la note de l'élève \n");
scanf ("%d", &un_eleve.note);
write (A COMPLETER);
i = i + 1;
}
ret = lseek(A COMPLETER);
if (ret==-1)
perror ("prob lseek");
printf ("la nouvelle position est %d\n", ret);
i = 0;
while(i<4)
{
read (A COMPLETER);
printf ("le nom et la note de l'élève sont %s, %d\n",
un_eleve.nom,
un_eleve.note);
i = i + 1;
}
close(A COMPLETER);
return 0;
}
```