Chapitre 12

Les verrous de fichiers

Mécanismes de contrôle d'accès concurrents à un fichier, les verrous sont d'une grande utilité dans les applications de gestion et dans l'élaboration de bases de données partagées.
Les verrous sont rattachés aux **inœuds**. Ainsi toutes les ouvertures d'un même fichier, et à fortiori tous les descripteurs sur ces ouvertures, "voient" le verrou.
La protection réalisée par le verrou a donc lieu sur le fichier physique.
Un verrou est la **propriété** d'un seul **processus**, et seul le processus propriétaire du verrou peut le modifier ou l'enlever, attention le verrou ne protège pas contre les accès du processus propriétaire (attention à une situation multi-thread).

12.1 Caractéristiques d'un verrou

Les verrous sont définis par deux caractéristiques :

La portée : Ensemble des positions du fichier auxquelles le verrou s'applique. Cet ensemble est un intervalle, soit une portion du fichier

`[position1, position2]`

soit jusqu'à la fin du fichier

`[position1, fin de fichier[`

dans ce dernier cas si le fichier augmente, le verrou protège les nouvelles positions.

Le type : qui décrit les possibilités de cohabitation des différents verrous.

F_RDLCK partagé, plusieurs verrous de ce type peuvent avoir des portées non disjointes, par exemple les verrous $[80, 150]$ et $[100, 123]$

F_WRLCK exclusif, pas de cohabitation possible avec un autre verrou quelque soit son type.

12.2 Le mode opératoire des verrous

Le mode opératoire joue sur le comportement des primitives `read` et `write`. Les verrous d'un fichier sont soit **consultatifs**, soit **impératifs** (NON-POSIX) [1] .

Dans le premier mode **advisory** (consultatif), la présence d'un verrou n'est testée qu'à la pose d'un verrou, la pose sera refusée s'il existe un verrou de portée non disjointe et que l'un des deux verrous est exclusif.

[1] En effet le mode impératif n'est pas POSIX, et donc par défaut n'est pas mise en oeuvre sur les disque sous linux.

Dans le second mode **mandatory**, la présence de verrous est testée pour la pose mais aussi pour les appels systèmes `read` et `write`.

Dans le mode consultatif, les verrous n'ont d'effet que sur les processus jouant effectivement le jeu, c'est-à-dire, posant des verrous sur les zones du fichiers sur lesquels ils veulent réaliser une lecture (verrou partagé) ou une écriture (verrou exclusif).

Dans le mode impératif, les verrous ont un impact sur les lectures/écritures de tous les processus :
- sur les verrous de type partagé (F_RDLCK), toute tentative d'écriture (appel système write) par un autre processus est bloquée ;
- sur les verrous de type exclusif (F_WRLCK), toute tentative de lecture ou d'écriture (read et write) par un autre processus est bloquée.

Pour rendre l'utilisation impérative il faut sous **linux** monter le disque avec l'option `-o mand`. Puis il faut utiliser la commande chmod pour positionner le SETGID bit, soit chmod g+s fichier en shell soit la même chose en C : si l'on a le descripteur `d` d'une ouverture sur le fichier

```
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
...
struct stat buf
fstat(d, &buf);
fchmod(d,buf.st_mode |  S_ISGID);
```

même chose avec stat et chmod si l'on la référence du fichier.

12.3 Manipulation des verrous

La structure de verrou `flock` :

```
struct flock  {
      short    l_type;      /* F_RDLCK, F_WRLCK,F_UNLCK */
      short    l_whence;    /* SEEK_SET,SEEK_CUR,SEEK_END */
      off_t    l_start;     /* position relative a l_whence */
      off_t    l_len;       /* longueur de l'intervalle */
      pid_t    l_pid;       /* PID du processus propriétaire */
};
```

le champ `l_type`

F_RDLCK verrou partagé

F_WRLCK verrou exclusif

F_UNLCK déverrouillage

Les manipulations de verrous se font avec la primitive `fcntl`, c'est-à-dire par le biais d'un descripteur. Pour poser un verrou partagé, ce descripteur doit pointer sur une ouverture en lecture. De même, il faut un descripteur sur une ouverture en écriture pour un verrou de type exclusif
.

Pour décrire la portée du verrou que l'on veut poser, on utilise la même syntaxe que pour la primitive `lseek`, le début de l'intervalle est **whence+l_start** :
`l_whence` = SEEK_SET $\longrightarrow$ whence = 0
`l_whence` = SEEK_CUR $\longrightarrow$ whence = offset courrant
`l_whence` = SEEK_END $\longrightarrow$ whence = taille du fichier.

La longueur du verrou est définie par le champ `l_len`. Si cette valeur est nulle, le verrou va jusqu'à la fin du fichier (même si le processus change cette fin). Remarque : il est possible de poser un verrou dont la portée est supérieure à la taille du fichier.

Le champ `l_pid` contient le pid du processus propriétaire du verrou, ce champ est rempli par `fcntl` dans le cas d'un appel consultatif (`F_GETLK`).

12.4 Utilisation de `fcntl` pour manipuler les verrous

```
#include <sys/types.h>
#include <unistd.h>
#include <fcntl.h>
int fcntl(int desc, int commande, struct flock *verrou);
```

`fcntl` retourne 0 en cas de succès, ou -1 en cas d'echec.

Trois commandes possibles :

F_SETLK pose non bloquante

si il existe un verrou incompatible, `errno` a pour valeur EAGAIN

si l'on n'a pas les droits d'accès sur le fichier pour le type de verrou demandé, alors `errno` a pour valeur EACCES ;

si la pose du verrou crée une situation d'interblocage, alors `errno` a pour valeur EDEADLK.

F_SETLKW pose bloquante (Wait)

succès immédiat si il n'y a pas de verrou incompatible, ou succès une fois les verrous incompatibles levés.

si l'appel est interrompu, `errno` a pour valeur EINTR

si une situation d'interblocage est détectée, alors `errno` a pour valeur EDEADLK.

F_GETLK Test d'existence d'un verrou incompatible avec le verrou passé en paramètre (retour -1 sur des paramètres incorrects)

si il existe un tel verrou incompatible, alors la structure flock passée en paramètre est remplie avec les valeurs de ce verrou incompatible. Le champ `l_pid` indique alors l'identité du processus propriétaire de ce verrou incompatible.

sinon, la structure flock reste inchangée excepté le champ type qui contient F_UNLCK.

Attention, après un test d'existence qui nous informe de l'absence de verrou incompatible, nous ne sommes pas assuré qu'au prochain appel la pose de ce verrou soit possible, en effet un autre processus a peut-être posé un verrou incompatible entre-temps (cf. interblocages chapitre 13).