Synchronisation et communication entre processus

Mémoire partagée et sémaphores sous LINUX

Schéma lecteurs / rédacteurs

La mémoire vue par Linux

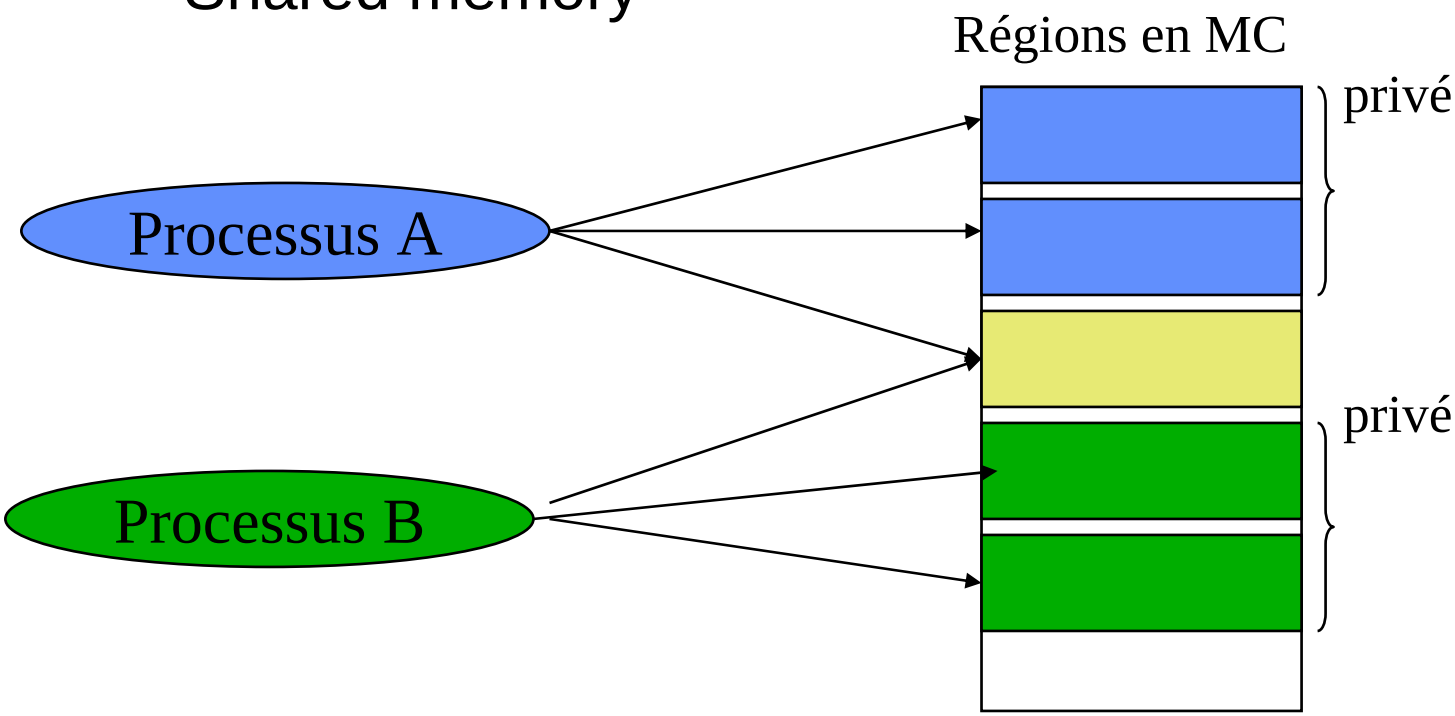
- L'espace d'adressage d'un processus est composé de régions
  - une région de code
  - une région des variables initialisées
  - une région des variables non initialisées
  - une région pour les codes et données des bibliothèques
  - une région pour la pile

- Une région est une zone contiguë de l'espace d'adressage traitée comme un objet pouvant être partagé et protégé. Elle est caractérisée par
  - ses adresses de début et de fin
  - les droits d'accès qui lui sont associés
  - l'objet qui lui est associé

- Une région est divisée en pages (4 Ko)

```
bbj>cat /proc/self/maps
08048000-0804a000 r-xp 00000000 03:02 7914
0804a000-0804b000 rw-p 00001000 03:02 7914
0804b000-08053000 rwxp 00000000 00:00 0
40000000-40005000 rwxp 00000000 03:02 18336
40005000-40006000 rw-p 00004000 03:02 18336
40006000-40007000 rw-p 00000000 00:00 0
40007000-40009000 r--p 00000000 03:02 18255
40009000-40082000 r-xp 00000000 03:02 18060
40082000-40087000 rw-p 00078000 03:02 18060
40087000-400b9000 rw-p 00000000 00:00 0
bfffe000-c0000000 rwxp fffff000 00:00 0
```

Shared memory

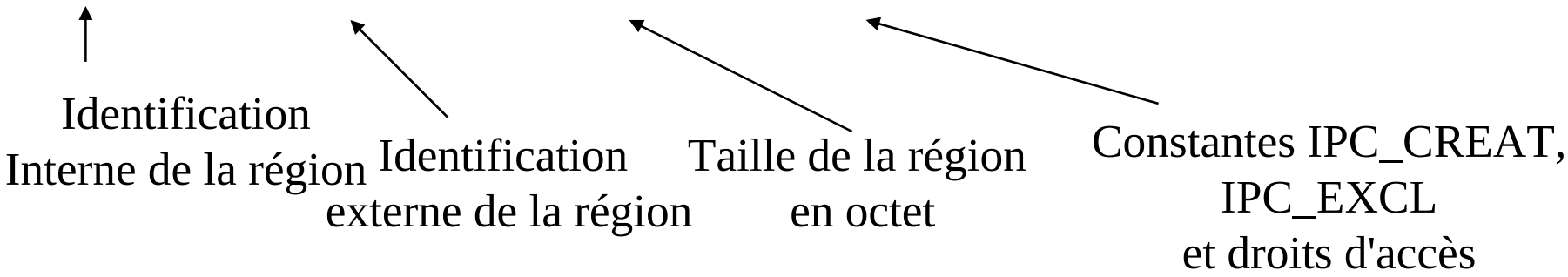


- Région de mémoire pouvant être partagée entre plusieurs processus
- Un processus doit attacher le région à son espace d'adressage avant de pouvoir l'utiliser
- Outil IPC repéré par une clé unique
- L'accès aux données présentes dans la région peut requérir une synchronisation (outil sémaphore)

Shared memory

- Création ou accès à une région de mémoire partagée

int shmget (key_t cle, int taille, int option)



- Attachement d'une région de mémoire partagée

void *shmat (int shmid, const void *shmadd, int option)

- Détachement d'une région de mémoire partagée

void *shmdt (void *shmadd)

Lecteurs /rédacteurs

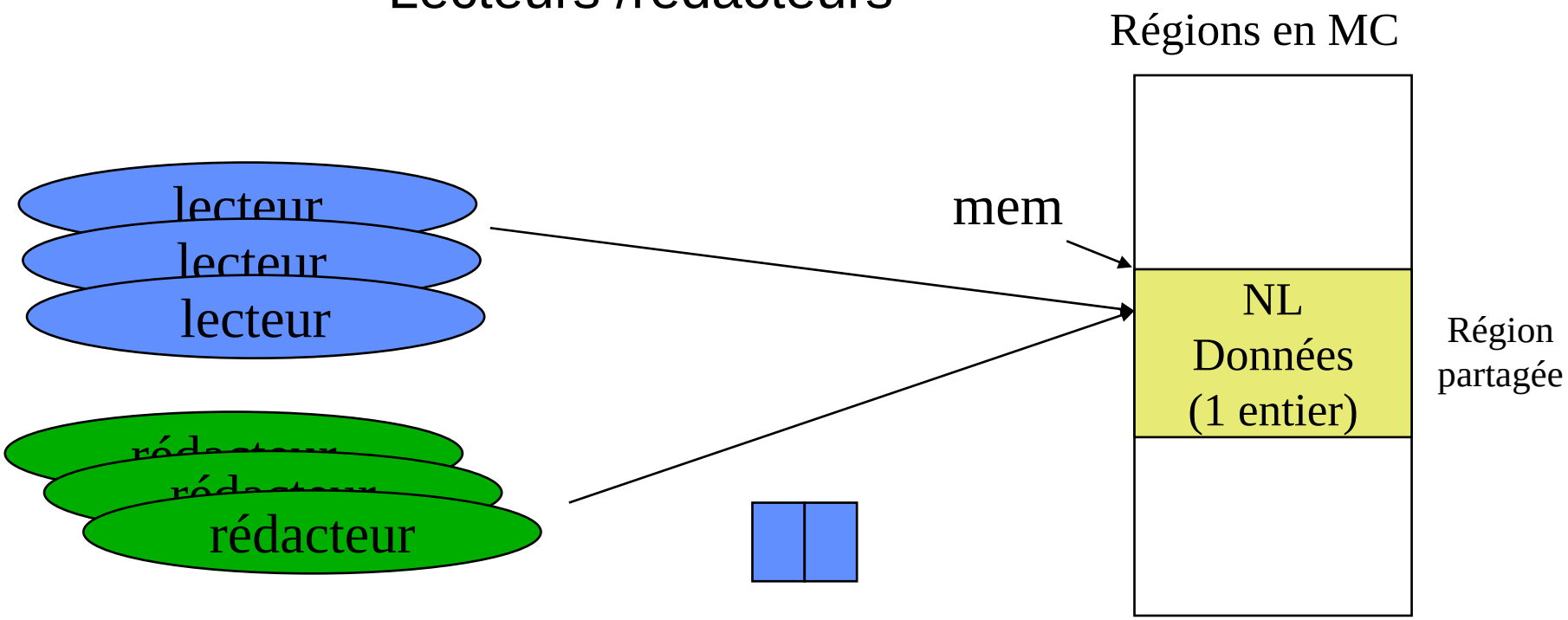


Tableau de deux sémaphores

Sembuf[0] = Accès

Sembuf[1] = Mutex

```
/*****************************************************************/
/*               processus principal                             */
/*****************************************************************/
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

#define CLE         256  /* cle du segment de mémoire */
#define CLES       300        /* clé du tableau de sémaphores */

main()
{
int shmid; semid, i;
char *mem ;
pid_t lecteur, redacteur;

/* création du segment de mémoire partagée avec la clé CLE */
shmid=shmget((key_t)CLE,1000,0750 |IPC_CREAT | IPC_EXCL);
/* attachement */
mem=shmat(shmid,NULL,0));
/*initialisation des variables NL et variable partagée par les lecteurs/rédacteurs*/
*mem = 0;
*(mem + 4) = 0;

/* création et initialisation des deux sémaphores MUTEX et ACCES */
semid = semget (CLES, 2, IPC_CREAT|IPC_EXCL|0600);
semctl (semid, 0, SETVAL, 1);
semctl (semid, 1, SETVAL, 1);
```

```
/*****************************************************************/
/*               processus principal                             */
/*****************************************************************/


/* création des processus lecteurs et rédacteurs */
i = 0;
while (i < 3) { lecteur = fork();
             if (lecteur == 0)  execl(“home/delacroi/lecteur”, “lecteur”, NULL);
             else
              {
               redacteur = fork();
                  if (redacteur == 0)  execl(“home/delacroi/redacteur”,
                                                “redacteur”, NULL);
                     else
                           i ++ }}
wait();

}
```

```
/*****************************************************************/
/*              processus rédacteur                              */
/*****************************************************************/
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

#define CLE         256
#define CLES        300

main()
{
int shmid; semid;
char *mem ;
struct sembuf *operation;

/* récupération du segment de mémoire partagée avec la clé CLE */
shmid=shmget((key_t)CLE,1000,0);
/* attachement */
mem=shmat(shmid,NULL,0));
/* récupération du tableau de sémaphores */
semid = semget ((key_t)CLES,2,0);

/* opération P (sémaphore ACCES est le premier élément du tableau */
operation.sem_num = 0;
operation.sem_op = -1;
operation.sem_flg = 0;
semop (semid, &operation, 1);
```

```
/*****************************************************************/
/*              processus rédacteur                              */
/*****************************************************************/

/* écriture dans le segment */
*(mem+4) = *(mem+4) * 10;



/* opération V (ACCES)*/
operation.sem_num = 0;
operation.sem_op =  1;
operation.sem_flg = 0;
semop (semid, &operation, 1);
}
```

```
/*****************************************************************/
/*              processus lecteur                                */
/*****************************************************************/
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

#define CLE         256
#define CLES        300

main()
{
int shmid; semid;
char *mem ;
int varlocale;
struct sembuf *operation;

 /* récupération du segment de mémoire partagée avec la clé CLE */
shmid=shmget((key_t)CLE,1000,0);
/* attachement */
mem=shmat(shmid,NULL,0));
/* récupération du tableau de sémaphores */
semid = semget ((key_t)CLES,2,0);

/* opération P(MUTEX) (sémaphore MUTEX est le deuxième élément du tableau */
operation.sem_num = 1;
operation.sem_op = -1;
operation.sem_flg = 0;
semop (semid, &operation, 1);
```

```
/*****************************************************************/
/*               processus lecteur                               */
/*****************************************************************/

/* incrément et test de NL */
*mem = *mem + 1;


if (*mem == 1) { /* P(ACCES) */

/* opération P (ACCES)*/
operation.sem_num = 0;
operation.sem_op =  -1;
operation.sem_flg = 0;
semop (semid, &operation, 1); }


/* opération V(MUTEX) (sémaphore MUTEX est le deuxième élément du tableau */
operation.sem_num = 1;
operation.sem_op = 1;
operation.sem_flg = 0;
semop (semid, &operation, 1);


}
```

```
/*****************************************************************/
/*               processus lecteur                               */
/*****************************************************************/

/* lecture dans le segment */
varlocale = *(mem+4);

/* opération P(MUTEX) (sémaphore MUTEX est le deuxième élément du tableau */
operation.sem_num = 1;
operation.sem_op = -1;
operation.sem_flg = 0;
semop (semid, &operation, 1);

/* décrément et test de NL */
*mem = *mem - 1;

if (*mem == 0) { /* V(ACCES) */
/* opération V (ACCES)*/
operation.sem_num = 0;
operation.sem_op =  1;
operation.sem_flg = 0;
semop (semid, &operation, 1); }

/* opération V(MUTEX) (sémaphore MUTEX est le deuxième élément du tableau */
operation.sem_num = 1;
operation.sem_op = 1;
operation.sem_flg = 0;
semop (semid, &operation, 1);

}
```