Université de Bretagne Sud
UFR SSI, Dept MIS
SYSTEME D'EXPLOITATION
CONCURRENCE
TP : Les sémaphores IPC

Email : Luc.Courtrai@univ-ubs.fr

Producteur Consommateur

Deux processus se partagent un même fichier.

- le fichier contient N entiers ; N étant limité à 10. Le fichier est géré comme un tampon et un entier supplémentaire en début de fichier donne le nombre d'entiers réels dans fichier.
- Un des processus, *le producteur,* dépose les entiers un par un dans le fichier, Le producteur génère des entiers dans l'ordre croissant (0,1,2,3,4 . . .49).
- Le deuxième processus, *le consommateur*, extrait l'information (le consommateur).

En utilisant les sémaphores IPC , écrire en C le programme producteur consommateur. Les deux processus sont créés par un processus père qui de plus initialisera le fichier et les sémaphores puis à la mort de ses fils les détruira.

Pour vous aider, reprenez la version du multif.c utilisant le sémaphores IPC, ainsi que le bout code permettant de stocker un tableau d'entiers dans un fichier.

Producteur Consommateur V2

A partir de l'exercice précédant, découper le programme en 4 fichiers exécutables : initialisation.c, producteur.c, consommateur.c, clean.c (dont avec 4 main()).

- initialisation.c : initalisae le fichier d'entier et les sémaphores.
- producteur.c : code du producteur.c
- consommateur.c : code du consommateur.c
- clean.c : arrêt de l'application et supprime les sémaphores.

Annexe 1 : fic2Tab et tab2fic

```
// Lecture écriture d'un tableau d'entiers à partir d'un fichier
// La tableau possède une taille réelle MAX mais ne peut contenir
// que N entiers (taille effective)
//
// Le premier entier du tableau (indice 0) doit contenir le nombre d'éléments
// effectifs  du tableau.
// Le premier élément est donc à l'indice 1.
//

// fic2tab
// charge un tableau à partir d'un fichier
// size :  la taille réelle  du tableau (tab[0] non compris)
//
int fic2tab(char * pathname,int * tab,int size){
```

```
  int cible;
   // lecture  du fichier
   if ( (cible  = open(pathname,O_RDONLY)) < 0){
     fprintf(stderr,"probleme d'ouverture du fichier\n");
     return -1;
   }

   if (read(cible,tab,(size+1) * sizeof(int)) !=(size+1) * sizeof(int)) {
     fprintf(stderr,"probleme de lecture du fichier\n");
     return -1;
   }
   close(cible);
   return 0;
}


// tab2 fic
// ecrit un tableau dans un fichier
// size :  la taille réelle du tableau (tab[0] non compris)
//

int tab2fic(char * pathname,int * tab,int size){
  int cible;
  // creation du fichier
   if ( (cible  = open(pathname,O_WRONLY|O_CREAT|O_TRUNC,0666)) < 0){
     fprintf(stderr,"probleme d'ouverture du fichier\n");
     return -1;
   }

   if (write(cible,tab,(size+1) * sizeof(int)) !=(size+1) * sizeof(int)) {
     fprintf(stderr,"probleme d'ecriture du fichier\n");
     return -1;
   }
   close(cible);
   return 0;
}
```

Annexe 2 : Les sémaphores IPC

fichier d'inclusion

```
#include <sys/sem.h>
```

```
int semget(key_t nom_ext,int nb, int acces)
```

Création d'un sémaphore
Le premier appel à la primitive
- crée un ensemble de *nb* sémaphores
- accès dont les droits d'accès (IPC_CREAT | 0666)
- la fonction retourne un nom interne de l'ensemble

Aux appels suivants la fonction retourne un nom interne de l'ensemble des sémaphores.

```
struct sembuf {
  unsigned short int sem_num; // numero de semaphore
  short              sem_op;  // numero d'operation
  short              sem_ftg; // option
}
```

```
int semop(int nom, struct sembuf *tab_op,int nb_op)
```

La fonction permet d'effectuer une liste d'opérations sur les sémaphores.
- nom : nom interne de l'ensemble
- nb_op : nombre d'opérations à effectuer (taille du tableau)
- tab_op : tableau des opérations

Définition d'une opération
- sem_num : numéro de sémaphore
- sem_op : type de l'opération (exemple +1 $->$ V(s) -1 $->$ P(s)
- sem_flg : option de l'opération

```
int semctl(int nom, int semnum, int op, union senum ARG);
```

Permet de controler un ensemble de sémaphores : lire de valeur x, les positionner, détruire l'ensemble des sémaphores ...
- nom : nom interne de l'ensemble
- semnum : numéro ou nombre de sémaphores
- op : type de l'opération
- args : options de l'opération par exemple le tableau des valeurs d'initialisation des sémaphores

type des opérations
- SETVAL initialise la valeur d'un sémaphore (semnum)
- GETVAL lit la valeur d'un sémaphore (semnum)
- SETALL initialise la valeur de l'ensemble des sémaphores
- GETALL lit la valeur de l'ensemble des sémaphores
- IPC_RMID détruit l'ensemble des sémaphores.