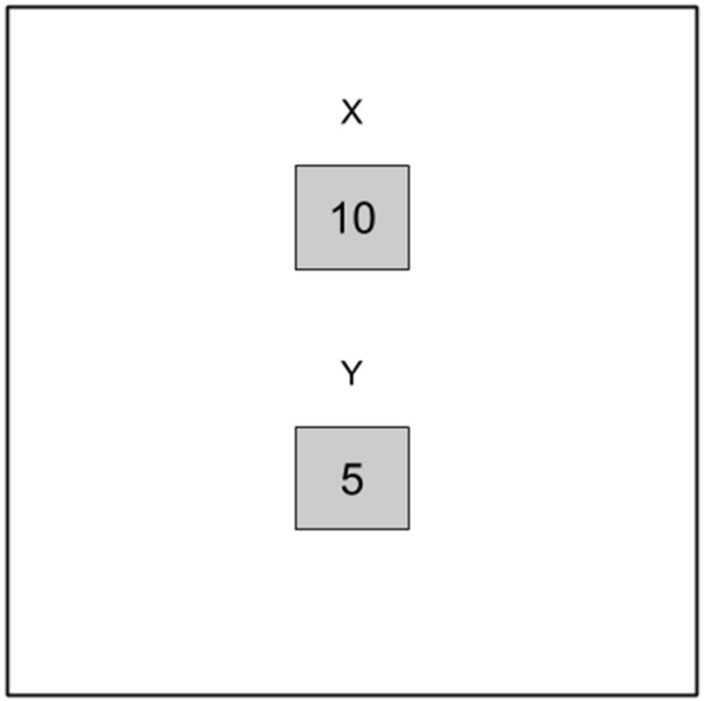
I2180
LINUX :
APPELS SYSTÈME

Mémoire partagée et Sémaphores

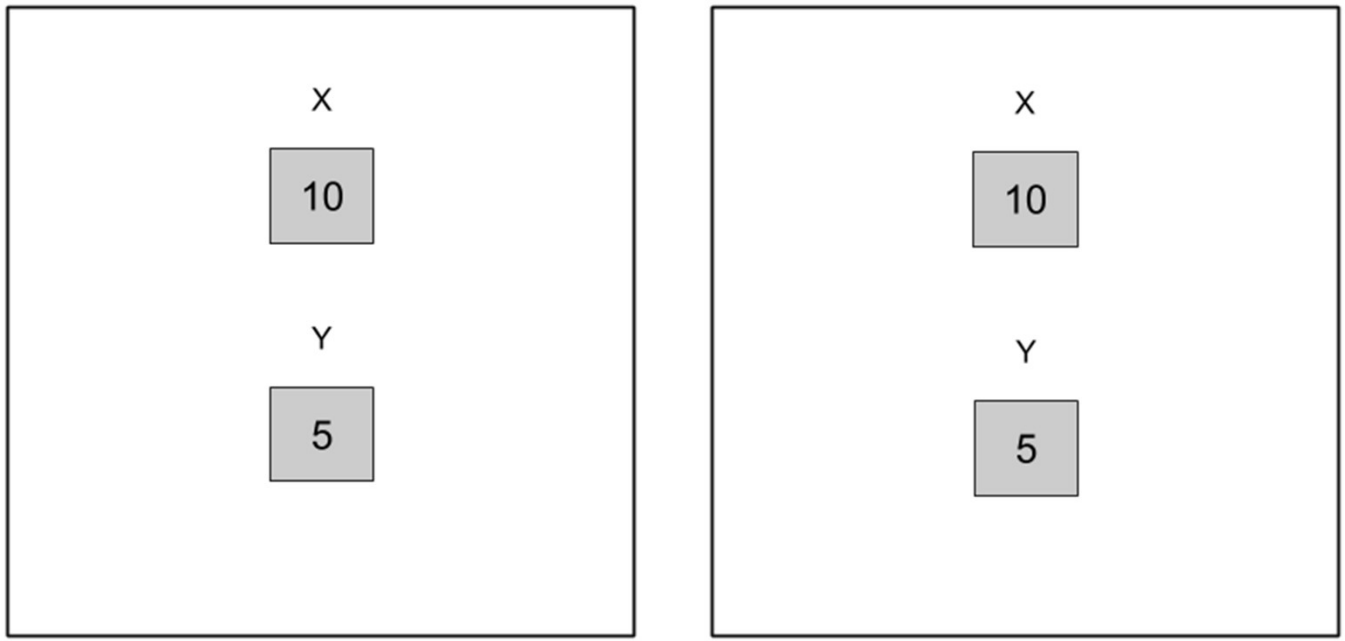
Semaines 06 et 07

José Vander Meulen

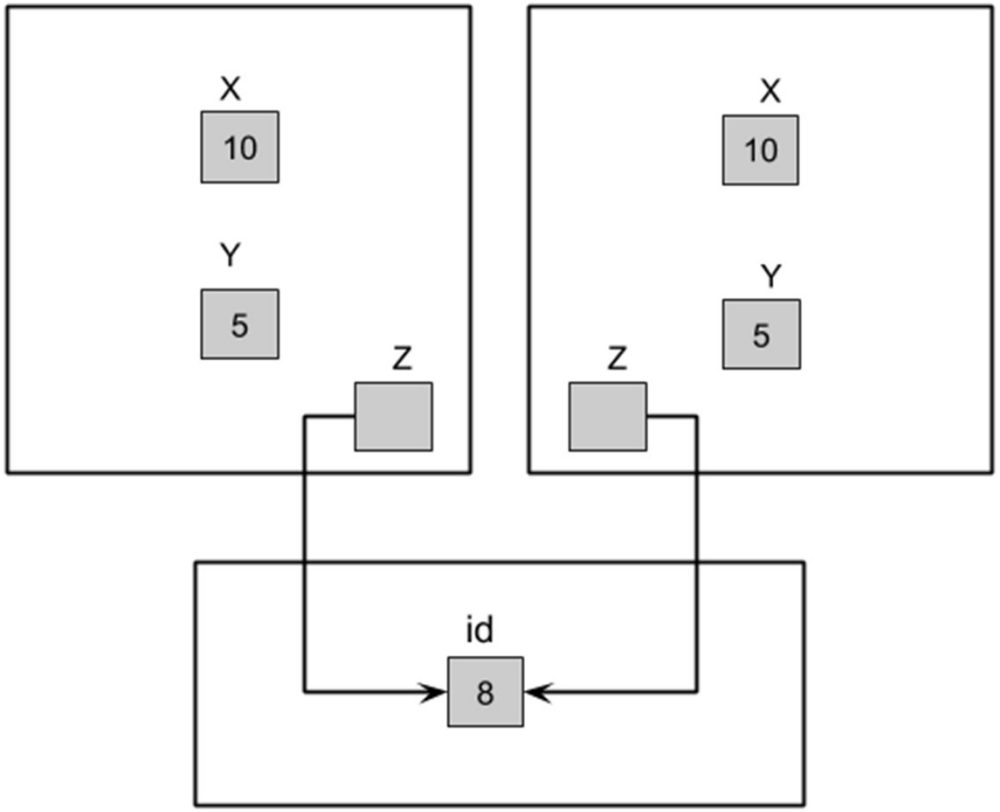
MÉMOIRE D'UN PROGRAMME



MÉMOIRE APRÈS UN FORK



MÉMOIRE PARTAGÉE



ÉTAPES

1. Récupérer identificateur (+ création du segment)
2. Attachement du segment
3. Détachement du segment
4. Contrôle du segment

RÉCUPÉRER IDENTIFICATEUR

int shmget(key_t key, size_t size, int shmflg)

Où

key : id

size : taille

shmflg : IPC_CREAT + IPC_EXCL + PERM

Renvoie

Un identificateur (cf autres méthodes) ou

-1 (si le segment est créé, initialisation à 0)

ATTACHEMENT DU SEGMENT

void* shmat(int shmid, void* a, int flg);

Où

shmid : identificateur

a : NULL dans le cadre de ce cours

flg : 0 dans le cadre de cours

Renvoie

Un pointeur vers mem partagée ou -1

DÉTACHEMENT DU SEGMENT

int shmdt(void* shmaddr);

Où

shmaddr : Un pointeur vers mem partagée

Renvoie

0 ou -1

CONTRÔLE DU SEGMENT

int shmctl(int shmid, int cmd, struct shmid_ds *buf);

Où

shmid : identificateur

cmd : IPC_RMID pour détruire le segment

buf : NULL dans le cadre de ce cours

Renvoie

0 ou -1

IPCS ET IPCRM

- Commande **ipcs**:

Info sur les IPCs de la machine

- Commande **ipcrm**:

Supprime des IPCS de la machine

EXAMPLE: SHMEX1.C

SÉMAPHORES

- Les sémaphores sont des ressources IPC
- Une variable entière est associée à chaque sémaphore, la valeur de cette variable n'est jamais négative
- Ils sont utilisés pour synchroniser des processus qui se partagent une ressource (cf. synchronized en Java)
- Une file d'attente est associée à chaque sémaphore. Elle contient la liste des processus souhaitant accéder à la ressource

DOWN ET UP

Deux opérations principales peuvent être effectuées sur un sémaphore :

- `down` (aussi appelé `proberen`) pour réserver, si possible, un accès à la ressource ;
- `up` (aussi appelé `verhogen`) pour rendre un accès à la ressource.

Le déroulement de ces opérations va être le suivant :

down		
si s strictement positif	→	`s--`
sinon	→	blocage du processus; mise du processus dans la file d'attente.

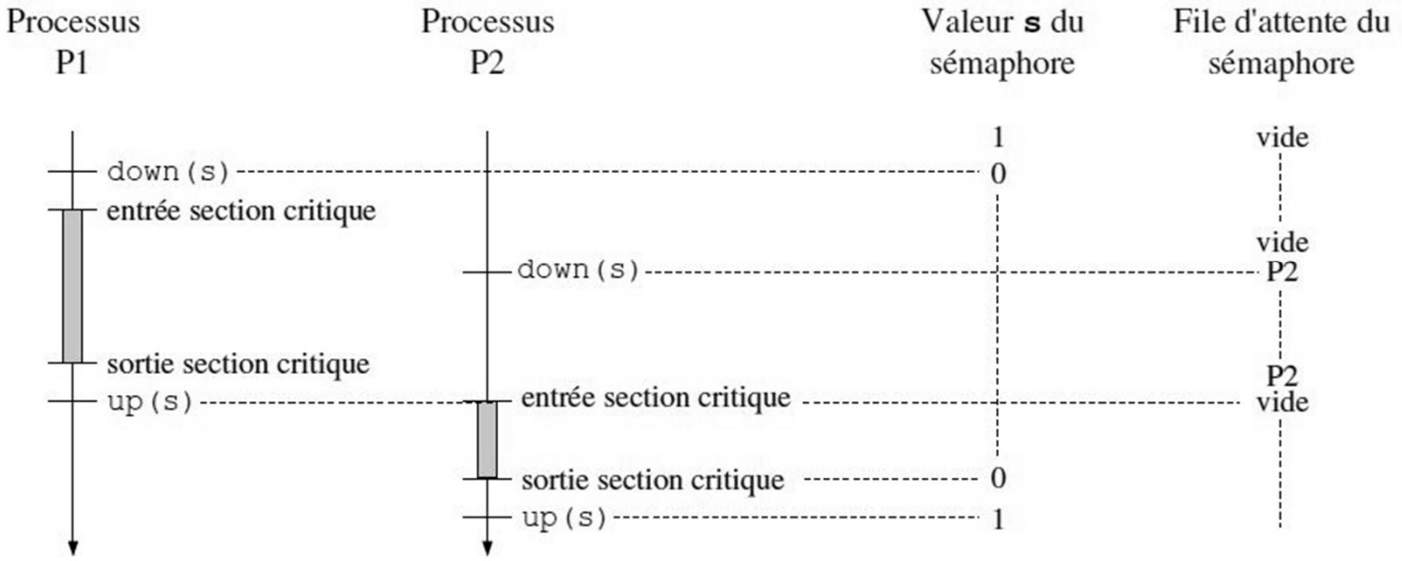
Déroulement de `down` dépend de s

up		
si file d'attente vide	→	`s++`
sinon	→	déblocage du premier processus de la file d'attente.

Déroulement de `up` dépend de la file d'attente

DOWN ET UP (EXEMPLE)

Exemple d'accès concurrent à une section critique par 2 processus, géré par un sémaphore **s**:



ÉTAPES

1. Création d'un ensemble de sémaphores
2. Opération sur les sémaphores
3. contrôle des sémaphores

RÉCUPÉRER IDENTIFICATEUR

int semget(key_t key, int nsems, int semflg);

Où

key : id

nsems : nombre de sémaphores

semflg : IPC_CREAT + IPC_EXCL + PERM

Renvoie

Un identificateur (cf autres méthodes) ou -1

ADDITION ET SOUSTRACTION

int semop(int id, struct sembuf* ops, size_t n);

Où

id : identificateur

ops : tableau d'opérations (cf example)

n : longueur du tableau d'opérations

Renvoie

0 ou -1

DESTRUCTION D'UN ENSEMBLE

int semctl(int semid, int semnum, int cmd);

Où

semid : identificateur

semnum : ignoré

cmd : IPC_RMID

Renvoie

0 ou -1

LIRE OU MODIFIER VALEUR

int semctl(int id, int n, int cmd, union semum arg); Où

id : identificateur

n : numéro de sémaphore

cmd : SET_VAL OU GET_VAL

arg : (cf example)

Renvoie 0 ou -1

EXAMPLE: SEMEX1.C

PLUS INFO

http://tinyurl.com/yyf85nnm

3.Les ressources IPC : sémaphore, mémoire partagée et files de messages

3.1.Objectifs

La séance a pour but de familiariser l'étudiant avec les sémaphores, la mémoire partagée et les files de messages. La manipulation des sémaphores nous servira à synchroniser des processus. La mémoire partagée, permettra le partage de données entre différents processus. Les files de messages sera un moyen plus pratique pour échanger des messages entre différents processus.

3.2.Généralités sur les sémaphores

Les sémaphores sont des objets de communication inter processus (IPC) utilisés pour synchroniser des processus. Une variable entière 's' est associée à chaque sémaphore et est une image du nombre de processus pouvant obtenir un accès à la ressource. Une file d'attente est également liée au sémaphore et reprend la liste des processus souhaitant un accès à la ressource.

Deux opérations principales peuvent être effectuées sur un sémaphore :

- `down` (aussi appelé `proberen`) pour réserver, si possible, un accès à la ressource ;
- `up` (aussi appelé `verhogen`) pour rendre un accès à la ressource.

Le déroulement de ces opérations va être le suivant :

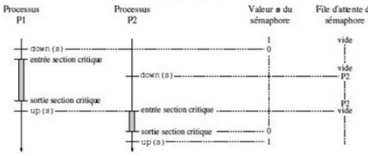
down		
si s strictement positif	→	s--
sinon	→	blocage du processus; mise du processus dans la file d'attente.

Déroulement de `down` dépend de s

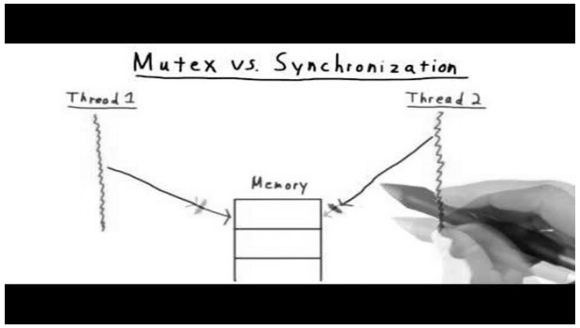
up		
si file d'attente vide	→	s++
sinon	→	déblocage du premier processus de la file d'attente.

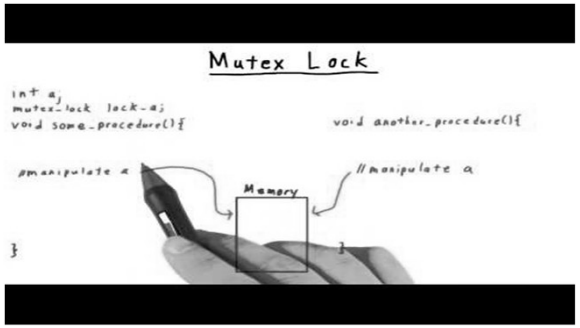
Déroulement de `up` dépend de la file d'attente

Exemple d'accès concurrent à une section critique par 2 processus, géré par un sémaphore **s**:



PLUS INFO


Mutex vs. Synchronization
Thread 1
Thread 2
Memory


Mutex Lock
int a;
mutex_lock lock_a;
void some_procedure(){
void another_procedure(){
//manipulate a
Memory
//manipulate a
}
}


Quick explanation:
Bounded-Buffer
(Producer-Consumer)
for Computer Science students