**SEMAFORI**

La semget definisce un ARRAY di più semafori

i**nt semget(key\_t chiave, int nsems, int semflg);**

**key\_t chiave** **= IPC\_PRIVATE;** oppure **key\_t chiave = ftok(".",'a');**

dove **IPC\_PRIVATE** crea una chiave che puo essere usata solo dal padre e dai figli, mentre **ftok();** genera una chiave usabile anche da processi di terzi, dove **"."** indica il path corrente e **'a'** indica una lettera causuale (nota bene i processi che vorranno usare la stessa risorsa dovranno generare la chiave in ugual modo mettendo lo stesso path uguale per tutti e la stessa lettera).

i**nt nsems** : numero di semafori da generare.

**int semflg** : **IPC\_CREAT | 0664,** flag piu permessi

**La semget();** restituisce l'id del semaforo che per comodità chiameremo id\_semaforo.

**int semctl(int id\_smaforo, int semnum, int cmd);**

**id\_semaforo** : l'identificatore della struttura semaforica creta tramite semget().

**semnum** : il numero di semaforo a cui vogliamo far riferimento

**int cmd** : il tipo di comando da fare **SETVAL, IPC\_RMID** dove *SETVAL* e il comando che ci dice che vogliao aggiungere un valore al semaforo esempi:

***semctl(id\_semaforo,0,SETVAL,1);***

***semctl(id\_semaforo,0,IPC\_RMID);***

***SEMAFORO STRUTTURA***

La semget definisce un ARRAY di più semafori. Ogni semaforo dell'array è una struttra dati che comprende:

**unsigned short semval;**  */\*valore del semaforo\*/*

**unsigned short semzcnt;** */\*num processi che aspettano 0\*/*

**unsigned short semncnt;**  /*\*num processi che aspettano incremento\*/*

**pid\_t sempid;**  */\*processo dell'ultima operazione\*/*

Su questi campi agisce semop (semaforo operazioni)

**int semop(int id\_semaforo, struct sembuf\* sops, unsigned nsops);**

ognuno degli nsops elementi, riferenti al puntatore sops, specifica un'operazione da compiere sul semaforo. L'operazione è descritta da una struttura, struct sembuf, la quale include i seguenti campi

**struct sembuf{**

**unsigned short sem\_num;** */\*numero di semaforo\*/*

**short sem\_op;** */\*operazione da compiere\*/*

**short sem\_flg;** */\*flags\*/*

**};**

**sem\_flg** : può assumere due valori **IPC\_NOWAIT o SEM\_UNDO**(sem\_undo annulera l'operazione una volta terminato il processo)

*i****nt semop(int id\_semaforo, struct sembuf\* sops, unsigned nsops);***

(ricordiamo che *id\_semaforo* e un identificativo della struttura semaforica e non del singolo semaforo)

Ogni operazione è eseguita sul semaforo individuato da ***sem\_num***(indica su quale semaforo tra quelli presenti nell'arrey, dovrà essere eseguita l'operazione). Il primo semaforo ha indice 0. Il valore specificato nel campo ***sem\_op*** nella *sembuf* specifica quale tipo d operazone effettuare:

***sem\_op < 0 : wait***

***sem\_op == 0 : wait\_for\_zero***

***sem\_op > 0 : signal***

**sem\_op > 0 Signal** : Se *sem\_op* è un intero positivo, l'operazione consiste nell'addizionare il valore di *sem\_op* al valore del semaforo(***semval*** – vedere struttura del semaforo).

***semval =+ sem\_op ;***

Al fine di eseguire loperazione di signal, il processo chiamante dovrà necessariamnete avere i permessi necessari alla modifica dei valori del semaforo, quest'operazione non causa il blocco del processo.

**sem\_op < 0 Wait** :Se *sem\_op* ha valore negativo, l'operazione si articoerà come di seguito:

1) se (**semval >= |sem\_op|** ) l'operazione procede immediatamente

2) se (**semval < |sem\_op|** ) il valore del campo ***semnct*** (- *vedere struttura semaforica*), viene incrementato di uno. Il processo si sospende. Per effettuare quest'operazione il processo chiamante deve avere i permessi per modificare il semaforo. Se è specificato nel falg **IPC\_NOWAIT** la system call fallisce(***errno = EAGAIN***), altrimenti esegue quello descritto sopra.

Nel caso 2) il processo sarà sospeso nell'attesa del verificarsi di una delle seguenti condizioni:

1) (**semval >= |sem\_op|** ), quando qiesta condizione sarà verificata il valore di ***smnct*** ( -vedere struttura semaforica ) sarà decrementato e il valore del semaforo srà modificato come segue:

***semval -= |sem\_op|;***

2) Il semaforo è rimosso: la system call fallisce (***errno = EAGAIN***).

**sem\_op == 0 Wait\_for\_zero** : Se *sem\_op* (- ***sembuf***) ha valore nullo, l'operazione specficata(*wait\_for\_zero*) è articolata nei seguenti passi:

1) Se **semval == 0**, l'operazione procede immediatamente (il processo non si sospende).

2) Altrimenti (**semval != 0** ) si procede come di seguito:

I) se specificato falg **IPC\_NOWAIT** in sam\_flg ( *- sembuf*), la system call fallisce (***errno = EAGAIN***).

II) Altrimenti la variabile **semzcnt** (- *vedere struttura semaforica* ) è incrementa di uno forzando il processo a sospendersi finché una delle seguenti condizioni non si verifica:

a) **semval == 0** (***semzcnt*** *è decrementato*). b) Il semaforo è rimosso : la system call fallisce (***errno = EAGAIN***).

Al fine di eseguire l’operazione di *“wait\_for\_zero”*, il processo chiamante dovrà almeno avere i permessi in lettura dei valori del semaforo.

**Rimozione di una struttura semaforica**

***semctl(id\_semaforo, num\_semaforo, IPC\_RMID);***

la variabile num\_semaforo in questo caso viene ignorata.

**Implementazioni di wait e signal**

**void** waitSem**(int** id\_semaforo**, int** num\_semaforo**){**

**struct sembuf** sem\_buf**;**

sem\_buf.sem\_num = num\_semaforo;

sem\_buf.sem\_flg = 0;

sem\_buf.sem\_op = -1;

**semop**(id\_semaforo, &sem\_buf, 1); /\**semaforo rosso*\*/

**}**

**void** waitSem**(int** id\_semaforo**, int** num\_semaforo**){**

**struct sembuf** sem\_buf**;**

sem\_buf.sem\_num = num\_semaforo;

sem\_buf.sem\_flg = 0;

sem\_buf.sem\_op = 1;

**semop**(id\_semaforo, &sem\_buf, 1); /\**semaforo verde\**/

**}**

**Creazione ed inizializzazione di un semaforo**

**...**

key\_t chiave\_sem = IPC\_PRIVATE;

// *richiesta di 2 semafori ed inizializzazione*

// il 2 sta ad indicare il numero di semafori da creare

id\_semaforo = **semget**( chiave\_sem, 2, IPC\_CREAT | 0664 );

// *Inizializzazione dei due semafori siccome erano due e semget() fa tornare un array li* //*inizializzo specificando la posizione che hanno nell'array il pirmo è in 0 il secondo in 1*

**semctl**(id\_semaforo, 0, SETVAL, val1 );

**semctl**(id\_semaforo, 1, SETVAL, val2);

...