SEMAFORI

La semget definisce un ARRAY di più semafori

int 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)

int 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***).

<u>sem_op == 0 Wait_for_zero</u> : 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);

        ...
```