Semafori POSIX

POSIX: Portable Operating System Interface for Unix (è una famiglia di standard IEEE)

I semafori POSIX consentono ai processi e ai thread di sincronizzare le loro azioni.

Un semaforo è un numero intero il cui valore non può mai scendere sotto lo zero.

Possono essere eseguite **due operazioni** sui semafori:

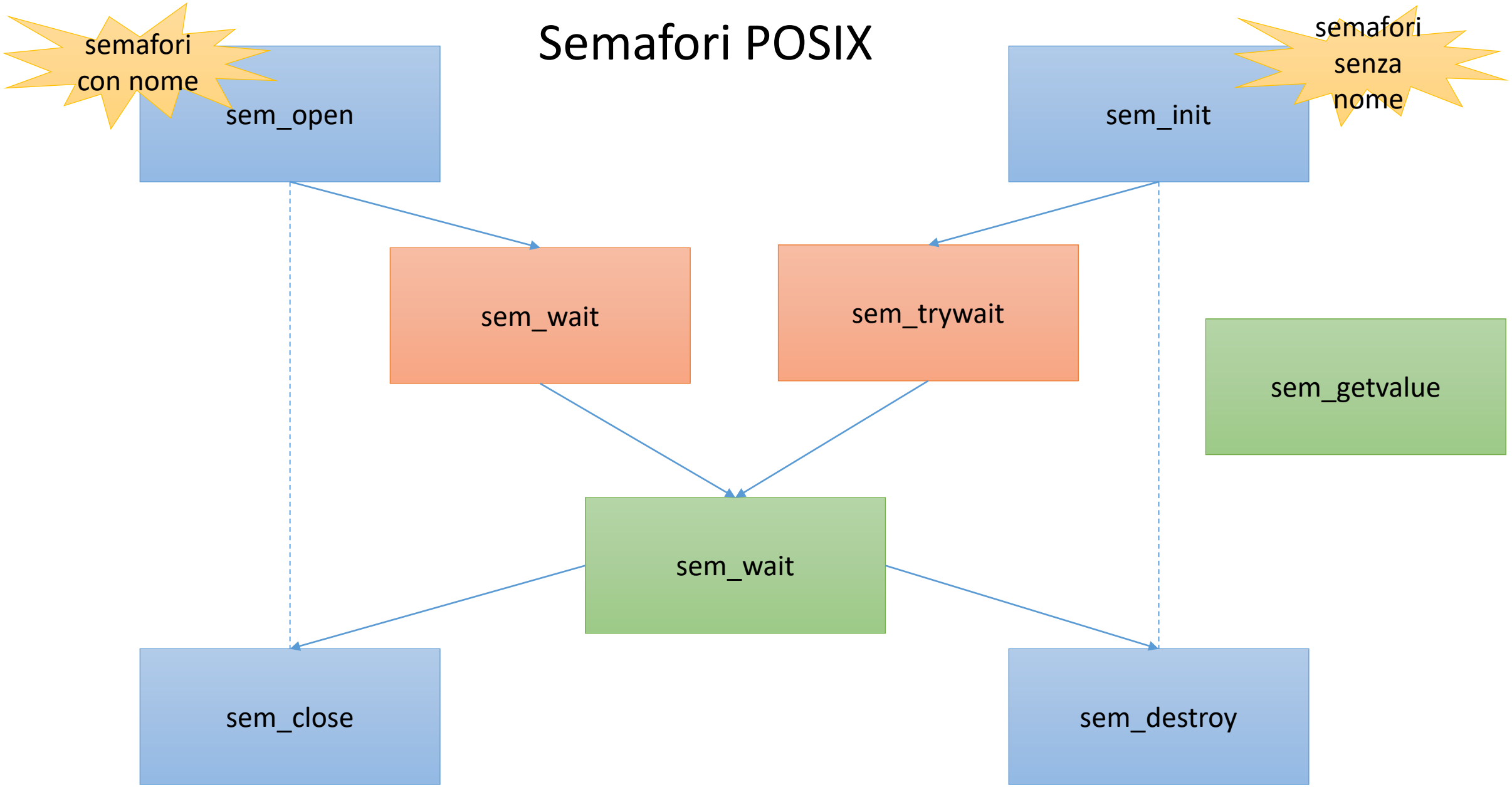
incrementa il valore del semaforo di uno (sem_post)

decrementa il valore del semaforo di uno (sem_wait).

Se il valore di un semaforo è attualmente zero, allora un'operazione sem_wait verrà bloccata fino a quando il valore diventa maggiore di zero.

I semafori POSIX si presentano in due forme: semafori con nome e semafori senza nome.

Semafori POSIX



sem_open

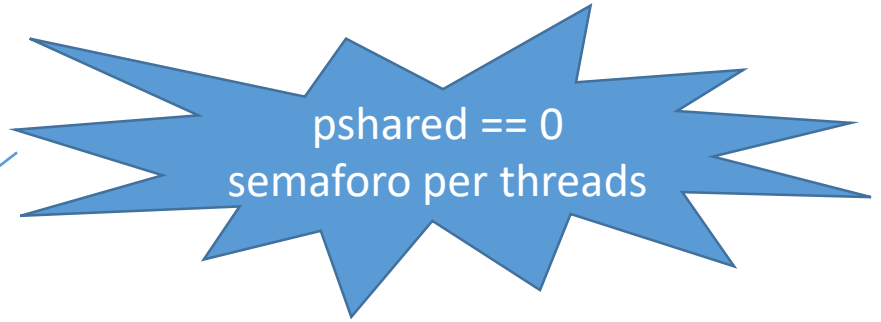
Crea un nuovo semaforo oppure apre un semaforo esistente, **con nome**.
sem_t *sem_open(const char *name, int oflag);
sem_t *sem_open(const char *name, int oflag, mode_t mode, unsigned int value);

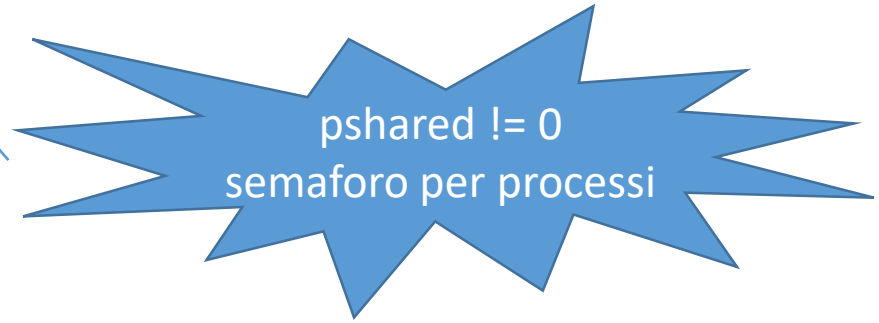
```
sem_t * result;
// se il named semaphore esiste, lo cancelliamo
sem_unlink(semaphore_name);
// valore iniziale del semaforo: 1
if ((result = sem_open(semaphore_name, O_CREAT, 00600, 1)) == SEM_FAILED) {
        perror("sem_open");
        return NULL;
}
```

sem_init

Inizializza un semaforo **senza nome**.

int sem_init(sem_t *sem, int pshared, unsigned int value);






```
sem_t * sem;
if (sem_init(sem, 1, 10) == -1) {
        perror("sem_init");
        exit(EXIT_FAILURE);
}
```

sem_wait

sem_wait () decrementa (blocca) il semaforo.

Se il valore del semaforo è maggiore di zero, il decremento procede e la funzione ritorna immediatamente.

Se il semaforo ha attualmente il valore zero, allora la chiamata si blocca fino a quando diventa possibile eseguire il decremento (cioè il valore del semaforo sale sopra lo zero) o un gestore di segnale interrompe la chiamata.

int sem_wait(sem_t *sem);

sem_wait

```
if (sem_wait(semaphore) == -1) {
        perror("sem_wait");
        exit(EXIT_FAILURE);
}

// vedere esempio 16sem_signal
// sem_wait potrebbe essere interrotta da un segnale il cui handler è stato impostato senza flag SA_RESTART
while ((res = sem_wait(semaphore)) == -1 && errno == EINTR)
        continue;
if (res == -1) {
        perror("sem_wait");
        exit(EXIT_FAILURE);
}
```

sem_post

sem_post incrementa (sblocca) il semaforo.

Se un altro processo o thread fosse bloccato in una chiamata sem_wait sullo stesso semaforo, il processo o thread verrà risvegliato e toccherà a questo bloccare a sua volta il semaforo.

```
if (sem_post(semaphore) == -1) {
        perror("sem_post");
        exit(EXIT_FAILURE);
}
```

sem_trywait

sem_trywait è uguale a sem_wait, tranne per il fatto che se il decremento non può essere eseguito immediatamente, la chiamata restituisce un errore (errno == EAGAIN) anziché bloccare l'esecuzione.

int s = **sem_trywait**(sem); // restituisce -1 ed errno == EAGAIN se il semaforo vale zero

```
if (s == -1 && errno == EAGAIN) {
    // il semaforo valeva zero
    ....
}
```

sem_timedwait

sem_timedwait è uguale a sem_wait, con la differenza che si specifica fino a quando può durare il blocco della chiamata se il decremento non può essere immediatamente eseguito. Il parametro indica un tempo assoluto rispetto a Epoch (1970-01-01 00:00:00 +0000 UTC).

Se il timeout è scaduto e il semaforo è ancora bloccato, la chiamata restituisce -1 e setta errno a ETIMEDOUT.

sem_getvalue

Restituisce il valore corrente del semaforo.

sem_getvalue(sem, &sem_val);

```
// dopo aver letto il valore del semaforo e prima del printf...
// il valore del semaforo potrebbe essere già cambiato
printf("prima di sem_trywait1 [%d] sem_val=%d\n", getpid(), sem_val);
```

sem_close, sem_destroy, sem_unlink

- Un semaforo con nome, aperto con sem_open, va chiuso con **sem_close**.
- Un semaforo con nome, creato con sem_open, va cancellato con **sem_unlink**.

- Un semaforo senza nome, inizializzato con sem_init, va distrutto con **sem_destroy**.

Sezione critica con semaforo

```
while ((res = sem_wait(semaphore)) == -1 && errno == EINTR)
        continue;
if (res == -1) {
        perror("sem_wait");
        exit(EXIT_FAILURE);
}

// SEZIONE_CRITICA

if (sem_post(semaphore) == -1) {
        perror("sem_post");
        exit(EXIT_FAILURE);
}
```