Лабораторная 14

Семафор - это целое число, значение которого никогда не должно опускаться ниже нуля. Над семафорами могут быть выполнены две операции: увеличьте значение семафора на единицу (sem_post(3)); и уменьшите значение семафора на единицу (sem_wait(3)). Если значение семафора в данный момент равно нулю, то операция sem_wait(3) будет блокироваться до тех пор, пока значение не станет больше нуля.

Используемые функции :

Сигнатура функции :

```
#include <semaphore.h>

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

Link with -pthread.
```

sem_init() инициализирует неименованный семафор по адресу на что указывает первый аргумент.

Аргументы :

Второй аргумент pshared указывает, должен ли этот семафор использоваться совместно между потоками процесса или между процессами.

- если он == 0, то семафор разделяется между потоки процесса и должны располагаться по некоторому адресу который виден всем потокам (например, глобальная переменная или переменная, динамически выделяемая в куче).
- Если pshared ≠ 0 то семафор совместно используется между процессами и должен располагаться в области общей памяти (смотрите shm_open(3), mmap(2) и shmget(2)). (С детства созданный fork(2) наследует сопоставления памяти своего родителя, он также может получить доступ к семафору.) Любой процесс, который может получить доступ к области общей памяти, может работать с семафором, используя sem_post(3), sem_wait(3) и так далее.

Последний аргумент задает начальное значение для семафора.

Возвращаемое значение :

0 - успех

-1 и errno выставляется - ошибка

Возможные ошибки :

EINVAL начальное значение превышает SEM_VALUE_MAX.

ENOSYS pshared отличается от нуля, но система не поддерживает общие для процесса семафоры (см. sem_overview(7)).

Сигнатура функции :

```
#include <semaphore.h>

int sem_wait(sem_t *sem);
int sem_trywait(sem_t *sem);
int sem_timedwait(sem_t *restrict sem,
                  const struct timespec *restrict abs_timeout);

Link with -pthread.
```

sem_wait() уменьшает значение(блокирует) передаваемого семафора.

Если значение семафора перед вызовом функции > нуля, то выполняется декремент и функция немедленно возвращается. Если семафор в данный момент имеет значение == 0, то вызов блокируется до тех пор, пока либо не станет возможным выполнить уменьшение (т.е. значение семафора не поднимется выше нуля), либо обработчик сигнала не прервёт вызов.

Возвращаемое значение :

0 - успех

-1 - при ошибке, значение значение семафора остается неизменным, и устанавливается значение errno для указания ошибки.

Возможные ошибки :

EINTR Вызов был прерван обработчиком сигнала

EINVAL семафор переданный первым аргументом является невалидным

Сигнатура функции :

```
#include <semaphore.h>

int sem_post(sem_t *sem);

Link with -pthread.
```

sem_post() инкрементирует семафор (разблокирует) , на который указывает sem.

Если значение семафора после вызова функции становится больше нуля, затем другой процесс или поток, заблокированный в вызове sem_wait(3), будет разбуженным и приступит к блокировке семафора

Возвращаемое значение :

0 - успех

-1 - при ошибке значение семафора остается неизменным, а значение errno устанавливается для указания ошибки.

Возможные ошибки :

EINVAL семафор переданный первым аргументом является невалидным

EOVERFLOW при вызове функции было бы превышено максимально допустимое значение для семафора

sem_post делать не на тот семафор, на который был сделан sem_wait

sem_post и sem_wait можно делать несколько раз одним потоком без предварительной блокировки/разблокировки

Завершив работу с неименованным семафором, его можно удалить вызовом функ-
ции sem_destroy.

NOTES top

sem_post() is async-signal-safe: it may be safely called within a signal handler.

Сигнатура функции :

```
#include <semaphore.h>

int sem_destroy(sem_t *sem);

Link with -pthread.
```

sem_destroy() уничтожает неименованный семафор по адресу на что указывает первый аргумент (уничтожить состояние, связанное с семафором, на который указывает sp)

После вызова sem_destroy семафор нельзя больше использовать для вызова функ-
ций управления семафором — только после повторной его инициализации вызо-вом sem_init.

Семафор, который был инициализирован sem_init(3), должен быть уничтожен с помощью sem_destroy()

Неименованный семафор должен быть вызвать sem_destroy() до того, как память, в которой он находится, будет освобождена.

Возвращаемое значение :

0 - успех

-1 - ошибка, устанавливается errno для указания ошибки

Возможные ошибки :

EINVAL семафор переданный первым аргументом является невалидным