Лабораторная 16

Используемые функции:

Сигнатура функции:

```
#include <fcntl.h>           /* For O_* constants */
#include <sys/stat.h>        /* For mode constants */
#include <semaphore.h>

sem_t *sem_open(const char *name, int oflag);
sem_t *sem_open(const char *name, int oflag,
                mode_t mode, unsigned int value);

Link with -pthread.
```

sem_open() создает новый семафор POSIX или открывает существующий семафор. Семафор идентифицируется по имени.

Именованный семафор идентифицируется именем (формы / somename) являющимся строкой с нулевым завершением длиной до NAME_MAX-4 (т.е. 251) символа, состоящие из начального правонаклонного слеша, за которым следует один или несколько символов, ни один из которых не является слешем. Два процесса могут работать с одним и тем же именованным семафором, передавая одно и то же имя в sem_open(3).

(

Имя не должно содержать других символов слеша, чтобы избежать любой зависимости от конкретных особенностей реализации. Например, если в используемой файловой системе имена /mysem и //mysem считаются одинаковыми, а данная конкретная реализация не использует файловую систему для поддержки семафоров, эти два имени могут интерпретироваться как разные (представьте, что произойдет, если реализация хеширует имена семафоров в целочисленные значения).

}

После открытия семафора им можно управлять с помощью sem_post(3) и sem_wait(3). Когда процесс завершил использование семафора, он может

использовать sem_close(3) для закрытия семафора. Когда все процессы завершат использование семафора, его
можно удалить из системы с помощью sem_unlink(3).

Доступ к именованным семафорам через файловую систему
В Linux именованные семафоры создаются в виртуальной файловой системе,
обычно монтируемой в /dev/shm, с именами вида
sem.somename. (Это причина, по которой имена семафоров
ограничены символами NAME_MAX-4, а не символами NAME_MAX.)

Аргумент oflag задает флаги, которые управляют работой вызова. (Определения значений флагов могут быть получены путем включения <fcntl.h> .) Если O_CREAT указан в oflag то создается семафор, если он еще не существует.

(Владелец (идентификатор пользователя) семафора устанавливается на действительный идентификатор пользователя вызывающего процесса. Принадлежность к группе (идентификатор группы) устанавливается на эффективный идентификатор группы вызывающего процесса.)

Если оба O_CREAT и O_EXCL указаны в oflag, затем возвращается ошибка, если семафор с указаннымименем уже существует.

O_EXCL принудительно создает файл. Если файл уже существует, вызов завершается ошибкой.

Он используется для обеспечения того, чтобы файл был создан с заданными разрешениями, переданными в третьем параметре. Короче говоря, у вас есть следующие варианты:

O_CREAT: Создайте файл с заданными разрешениями, если файл еще не существует. Если файл существует, он открывается, а разрешения игнорируются.

O_CREAT | O_EXCL: Создайте файл с заданными разрешениями, если файл еще не существует. Если файл существует, он завершается с ошибкой. Это полезно для того, чтобы создавать файлы блокировки и гарантировать эксклюзивный доступ к файлу (при условии, что все программы, использующие этот файл, следуют одному и тому же протоколу).

O_CREAT | O_TRUNC: Создайте файл с заданными разрешениями, если файл еще не существует. В противном случае сократите файл до нуля байт. Это дает больше эффекта, которого мы ожидаем, когда думаем "создать новый пустой файл". Тем не менее, он сохраняет разрешения, уже присутствующие в существующем файле.

Аргумент mode указывает разрешения, которые должны быть размещены на новом семафоре, как для open(2). (Символические определения битов разрешений можно получить, включив <sys/stat.h> .) Настройки разрешений маскируются под маску процесса umask. **Разрешение как на чтение, так и на запись должно быть предоставлено каждому классу пользователей, которые будут получать доступ к семафору.**

Окончательные права доступа к семафору определяются значением аргумента mode и маской процесса для прав доступа при создании файлов. Учитываются только права на чтение и на запись.

Аргумент value задает начальное значение для нового семафора. Если указан O_CREAT и семафор с указанным именем уже существует, то режим и значение игнорируются.

Возвращаемое значение:

успех - возвращает адрес нового семафора;

этот адрес используется при вызове других функций, связанных с семафором.

ошибка - SEM_FAILED, при этом для указания ошибки установлено значение errno.

Возможные ошибки:

EACCES семафор существует, но вызывающий не имеет прав, чтобы открыть его.

EEXIST И O_CREAT, и O_EXCL были указаны в oflag, но семафор с таким именем уже существует.

EINVAL value > SEM_VALUE_MAX.

EINVAL имя семафора состоит только из "/", за которым не следует никаких других

символов.

EMFILE Ограничение на количество открытых файлов для каждого процесса дескрипторы были достигнуты.

ENAMETOOLONG имя было слишком длинным.

ENFILE общесистемное ограничение на общее количество открытых файлов было достигнуто.

ENOENT Флаг O_CREAT не был указан в oflag и семафора с таким именем не существует; или O_CREAT был указан, но имя не было правильно сформировано.

ENOMEM Недостаточно памяти.

Сигнатура функции:

```
#include <semaphore.h>

int sem_close(sem_t *sem);

Link with -pthread.
```

sem_close() закрывает именованный семафор, на который ссылается переданный аргумент, позволяя освободить любые ресурсы, которые система выделила вызывающему процессу для этого семафора.

Возвращаемое значение:

успех - 0

ошикбка - -1 и errno установливается для указания на ошибку.

Возможные ошибки:

EINVAL переданный семафор не является допустимым семафором.

Сигнатура функции:

```
#include <semaphore.h>

int sem_unlink(const char *name);

Link with -pthread.
```

Функция sem_unlink удаляет имя семафора. Если в системе не остается открытых ссылок на семафор, он также удаляется. Иначе удаление семафора откладывается,
пока не будет закрыта последняя ссылка на него.

Возвращаемое значение:

успех - 0

ошибка - -1 и errno установливается для указания на ошибку.

Возможные ошибки:

EACCES У вызывающего нет разрешения на отсоединение этого семафор.

ENAMETOOLONG имя было слишком длинным.

ENOENT Нет семафора с заданным именем.

Сигнатура функции:

```
#include <unistd.h>

pid_t fork(void);
```

fork() создает новый процесс, дублируя вызывающий процесс.

Новый процесс называется дочерним процессом. Вызывающий процесс - родительский.

Указатели на именнованные семафоры остаются доступными и валидными так как после форка процесс копирует сегмента данных, стека и кучи. (используется

метод, который получил название ко-пирование при записи (copy-on-write, COW). Указанные выше области памяти ис-пользуются совместно обоими процессами, но ядро делает их доступными только для чтения. Если один из процессов попытается изменить данные в этих областях, ядро немедленно создаст копию конкретного участка памяти; обычно это «страни-
ца» виртуальной памяти.)

On success, the PID of the child process is returned in the parent, and 0 is returned in the child. On failure, -1 is returned in the parent, no child process is created, anderrno is set to indicate the error.

Сигнатура функции:

```
#include <semaphore.h>
#include <time.h>

int sem_timedwait(sem_t *restrict sem,
    const struct timespec *restrict abstime);
```

Функция sem_timedwait() должна заблокировать семафор, на который ссылается sem, как в функции sem_wait(). Однако, если семафор не может быть заблокирован без ожидания, пока другой процесс или поток разблокирует семафор, выполнив функцию sem_post() , это ожидание должно быть прекращено по истечении указанного времени ожидания.

Время ожидания истекает, когда проходит абсолютное время, указанное abstime, измеряемое часами, на которых основаны таймауты (то есть, когда значение этого времени равно или превышает abstime), или если абсолютное время, указанное abstime
, уже прошло во время вызова.

Время ожидания должно основываться на часах CLOCK_REALTIME. Тип данных timespec определяется как структура в заголовке<time.h> .

Ни при каких обстоятельствах функция не должна завершаться с таймаутом, если семафор может быть заблокирован немедленно. Нет необходимости проверять действительность abstime, если семафор может быть заблокирован немедленно.

(

Если значение семафора можно уменьшить немедленно, второй аргумент игнориру-
ется: даже если он будет содержать отметку времени в прошлом, попытка умень-шить значение семафора все равно будет выполнена.

)

Возвращаемое значение:

0 - вызывающий процесс успешно выполнил операцию блокировки семафора для семафора, обозначенного sem.

-1 и установить errno для указания ошибки - если вызов был неудачным, состояние семафора должно быть неизменным

Возможные ошибки:

EINVAL Процесс или поток был бы заблокирован, а параметр abstime
указывал значение поля наносекунд меньше нуля или больше или равно 1000 миллионов.

ETIMEDOUT
Не удалось заблокировать семафор до истечения указанного
времени ожидания.

Функциясem_timedwait() может завершиться с ошибкой, если:

EDEADLK
Было обнаружено состояние взаимоблокировки.

EINTR сигнал прервал эту функцию.

EINVAL семафор переднный как аргумент не ссылается на действительный семафор.

Получить время конкретных часов можно с помощью функции clock_gettime.

Сигнатура функции:

```
#include <time.h>

int clock_getres(clockid_t clockid, struct timespec *res);

int clock_gettime(clockid_t clockid, struct timespec *tp);

int clock_settime(clockid_t clockid, const struct timespec *tp);

Link with -lrt (only for glibc versions before 2.17).

Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

clock_getres(), clock_gettime(), clock_settime():
        _POSIX_C_SOURCE >= 199309L
```

Функции clock_gettime() и clock_settime() извлекают и устанавливают

время указанного clockid часов.

Все реализации поддерживают общесистемные часы реального времени, которые

идентифицируются с помощью CLOCK_REALTIME. Его время представляет собой секунды и

наносекунды с момента начала Эпохи. Когда его время изменяется, таймеры для

относительный интервал не изменяется, но затрагиваются таймеры для абсолютного момента времени

CLOCK_REALTIME

Настраиваемые общесистемные часы, которые измеряют реальные (т.е. настенные-

часы) время. Установка этих часов требует соответствующих привилегий-

легес. На эти часы влияют прерывистые скачки

системного времени (например, если системный администратор вручную

изменяет часы), а также путем постепенных корректировок за-

формируется с помощью adjtime(3) и NTP.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ вверху

clock_gettime(), clock_settime() и clock_getres() возвращают 0 для

успех, или -1 для сбоя (в этом случае устанавливается errno

соответствующим образом).

ОШИБКИ сверху

EFAULT tp указывает за пределы доступного адресного пространства.

EINVAL Указанный clockid недействителен по одной из двух причин.

Либо жестко закодированное положительное значение в стиле System-V находится вне

диапазона, либо динамический идентификатор часов не относится к допустимому

экземпляр объекта clock.

ENOTSUP

Операция не поддерживается указанным устройством синхронизации dynamic POSIX.

EINVAL (начиная с Linux 4.3)Вызов clock_settime() с идентификатором CLOCK_REALTIME

предпринята попытка установить время на значение, меньшее текущего

значение CLOCK_MONOTONIC clock.

ENODEV Устройство с возможностью горячего подключения (например, USB), представленное

динамическим идентификатором clk_id, исчезло после его символьного устройства

был открыт.

.