

Семафоры

Семильтные вызовы: `semget`, `semop`, `semctl`  
<sup>1</sup>      <sup>2</sup>      <sup>3</sup>

### 1. **semget()** — создаёт набор семафоров

- **Формат:**

- `key-t key` — идентификатор, получ. с помощью с.б. `ftok`.
- `int nsems` — количество семафоров в наборе
- `int semflg` — флаги.

- Возвращает идентификатор набора семафоров.

- Набор семафоров создаётся, если `key = IPC_PRIVATE` или `semflg` опр. как `IPC_CREAT`. и с ключом `key` не связан ни один набор семафоров.

! В `semflg`: `IPC_CREAT & IPC_EXCL` и набор семафоров уже существует → ошибка.

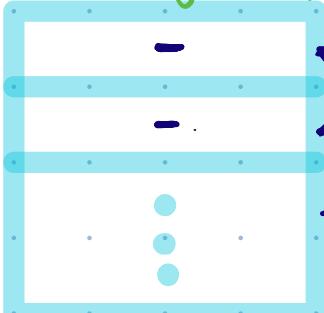
`IPC_EXCL` - exists control - контроль существования.

- Инициализирует структуру `semid_ds` →

`struct semid_ds { <sys/sem.h>`

```
struct ipc_perm sem_perm; // права доступа к набору семафоров
time_t sem_ctime; // время создания/последнего изм с помощью с.б. semctl
time_t semotime; // время последнего сем. бояса semop
unsigned long sem_nsems; // количество семафоров в наборе
```

→ Идентификатор, дескриптор набора семафоров



Лабилья наборов семафоров

дескриптор наборов семафоров

`struct semid_ds <sys/sem.h>`

## struct ipc\_perm {

key\_t key; // Ключ, переданный в semget  
uid\_t uid; // Эффективный UID владельца  
gid\_t gid; // Эффективный GID владельца  
uid\_t euid; // — — создатель  
gid\_t egid; // — — создатель  
unsigned short mode; // Права доступа  $\text{rwx}$  для user, group, other  
unsigned short -- seq; // Порядковой номер (?)

2. **semop()** - выполнение межпотоковых операций со всеми значениями или частью набора семафоров.

- Параметры:

- int semid; - идентификатор набора семафоров
- struct sembuf \*sops; - указатель на массив структур-операций для набора семафоров.
- size\_t nsops; - количество выполненных операций в sops.

- Семафор в System V - атомарная структура

unsigned short semval; // значение семафора  
unsigned short semzcnt; // # ожиданий обнуления  
unsigned short semnent; // # ожиданий увеличения  
pid\_t sempid; // PID процесса, который последний изм. счет. семаф.

- struct sembuf h

unsigned short semnum; // Номер семафора  
short sem\_op; // Операция на семафоре  
short sem\_flg; // Режим операции

- Определены 3 операции:

- 1) Декремент (-1): sem\_op < 0 - захват ресурса, если возможен, иначе блокировка.
- 2) Гарантия не 0: sem\_op == 0 - процесс переводится в состоянии блокировки без захвата семафора
- 3) Инкремент (+1): sem\_op > 0 - отменяет захват семафора, фрагмент. первого процесса впереди ожидания семаф.

- Флаги:

IPC\_NOWAIT } указывает системе о немедленном пром.  
sem undo } переходит в сис. ожидания свободн. семафора  
} указывает, что его должны отменяться измененные значения семафора и при заверш. проц. - ликвидир. изм-я

Процесс может аварийно завершиться если получит сигнал kill (невозможно перехватить)  $\Rightarrow$  не сможет освободить ресурс  $\Rightarrow$  ожидание этого процесса будет блокировано навсегда.

**sem - flg = 0** - операция sys sem Вополнение по умолчанию - без приоритетов  
 $\hookleftarrow$  IPC - NOWAIT / sem - UNDO

### • Гарантия

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>

struct sembuf sops[2] = { { 0, -1, SEM_UNDO | IPC_NOWAIT },
                           { 1, 0, 1 } }; ??

int main() {
    int perm = S_IRWXU | S_IRWXG | S_IROTH;
    int fd = semget(100, 2, IPC_CREATE | perm);
    if (fd == -1) perror("semget");
    if (semop(fd, sops, 2) == -1) perror("semop");
    return 0;
}
```

Первый семафор работает как гарантия, что другой процесс его освободит  $\Rightarrow$  SEM\_UNDO.

### 3. **semctl()** - управление набором семафоров.

#### • Гарантия:

- int semid - идент. набора семафоров
- int semnum - номер семафора
- int op - операции
- ...

? **sem - flg = 1** - содержит общий приоритет IPC\_NOWAIT.

## Семафор разделяемой памяти -

средство передачи данных между процессами, буфер - некий выделенный блок адресов, нач.адр и размер указены.

Создается в системной области памяти, т.к. взаимодействие процессов происходит через 3<sup>е</sup> адр. кр-во - All adra.

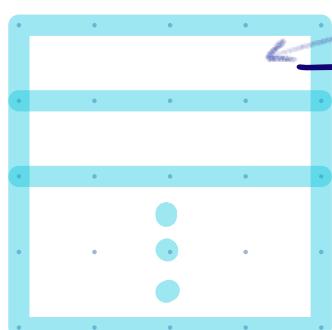
<sys/shm.h>

1. **shmget** - возвращает идентификатор разделяемой памяти.

- key + key - аналогично в semget()
- size + size - размер буфера
- int. flags - флаги.

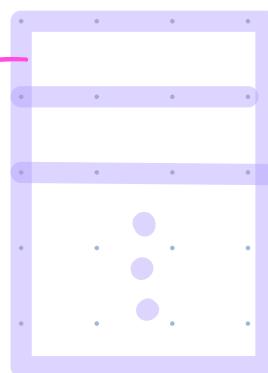
**IPC-PRIVATE** - создается новой сессии разделяемой памяти, идент. - свяж. с существующей с концептуальным key.

Нем. механизм встроенных средств взаимодействия.



дескриптор  
разделяемого семафора

Массив структур -  
мног. таблица.



свяж. список  
процессов

Мног.таблица сессий  
разделяемой памяти

**semat()** - присоединение

**sem d+()** - отсоединение

Каждый дескриптор это набор описываемых разрешений сегмента - указателей.

! Необходимо указать адрес.

$\text{MAX\_SIZE} = 2^{24}$ ,  $\text{MIN\_SIZE} = 1 \text{ байт}$ , но  $\text{PAGE\_SIZE}$  – зарезервированный минимальный размер, т.к. таблица будет размещаться страницами (4096)

или addr ег. памяти.

## Пример

```
#include <sys/shm.h>
int main() {
    int perm = S_IRWXU | S_IROKG | S_IWXO;
    int fd = shmget(100, 1024, IPC_CREAT | perm);
    if (fd == -1) perror("shmget");
    char * addre = (char *) shmat(fd, NULL, 0);
    if (addre == (char *) -1) perror("shmat");
    strcpy(addre, "aaa");
    if (shmat(addre) == -1) perror("shmat");
    return 0;
}
```

Уже есть 1<sup>й</sup> необходимый адрес.

Удаляемое разрешение сегмента с помощью предыдущего завоювання таємни.

## Лабораторная работа:

### 1) Приведение - потребление

Не менее 3<sup>х</sup> производителей и не менее 4<sup>х</sup> потребителей.  
Единиче замок - блокир. наст. опровергн.

### 2) Указатели - писатели

Не менее 3<sup>х</sup> писателей, не менее 5<sup>х</sup> читателей.

Процессы создают fork(), адреса указателей, много не писать (без писателя), слуз. дескрипторы.