

Средства System V IPC

Системные вызовы и библиотеки Unix SVR4

Иртегов Д.В.

ФФ/ФИТ НГУ

Электронный лекционный курс подготовлен в рамках реализации

Программы развития НИУ-НГУ на 2009-2018 г.г.

ЦЕЛИ РАЗДЕЛА

- иметь представление об общих свойствах
 - семафоров
 - разделяемой памяти
 - очередей сообщений
- использовать ключ для получения доступа к средствам IPC
- получать информацию о средствах IPC и удалять их с использованием команд

СРЕДСТВА МЕЖПРОЦЕССНОГО ВЗАИМОДЕЙСТВИЯ

- разделяемая память
- семафоры
- очереди сообщений

ОБЩИЕ СВОЙСТВА СРЕДСТВ IPC

- Средства могут быть созданы за любое время до их использования
- В момент создания для средства IPC устанавливаются собственность и права доступа, которые могут позднее быть изменены
- Содержимое средства IPC может сохраняться после того, как все использовавшие его процессы завершились.
- Средства должны удаляться явным образом, командой или соответствующим системным вызовом
- Средства IPC теряются при перезагрузке системы.

Страницы руководства IPC

msg	get	Используется для создания средства IPC, возвращает идентификатор средства.
sem		
shm		
msg	ctl	Определяет состояние, устанавливает различные опции и права доступа или удаляет идентификатор IPC
sem		
shm		
msg	op	Осуществляет операции над средством IPC. В действительности, это группы системных вызовов, например, msgsnd и msgrcv для очередей, shmat и shmdt для памяти.
sem		
shm		

struct ipc_perm

```
struct ipc_perm {
    uid_t uid; /* owner's user id */
    gid_t gid; /* owner's group id */
    uid_t cuid; /* creator's user id */
    gid_t cgid; /* creator's group id */
    mode_t mode; /* access modes */
    ulong seq; /* slot usage sequence number */
    key_t key; /* key */
    long pad[4]; /* reserve area */
};
```

get — основные сведения

- создатель задает `IPC_CREAT` и права доступа в параметре `flg`
- создатель может дополнительно задать `IPC_EXCL`
- существуют права чтения (`r`) и записи (`w`) для хозяина, группы и других пользователей.
- параметры настройки системы задают ограничения
- после создания средства `IPC`, устанавливается взаимно однозначное соответствие между ненулевым ключом и `id`
- `IPC_PRIVATE` предоставляет приватный ключ

ctl — основные сведения

IPC_STAT - получает информацию о состоянии

IPC_SET - изменяет хозяина или права доступа

IPC_RMID - удаляет средство

КОМАНДЫ `ipcs(1)` И `ipcrm(1)`

- состояние всех существующих в данный момент средств IPC - `ipcs(1)`
- удаление средства IPC - `ipcrm(1)`

ОЧЕРЕДИ СООБЩЕНИЙ

`msgget` - создать очередь или получить к ней доступ

`msgctl` - определить состояние очереди;
изменить хозяина или права доступа к ней;
изменить максимальный размер очереди
или удалить ее

`msgop` - послать (`msgsnd`) или получить (`msgrcv`) сообщение

msgget(2)

ИСПОЛЬЗОВАНИЕ

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

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

```
#include <sys/msg.h>
```

```
int msgget (key_t key, int msgflg);
```

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

успех - неотрицательное число, идентификатор
очереди сообщений

неуспех - -1 и errno установлена

msgctl(2)

ИСПОЛЬЗОВАНИЕ

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
int msgctl (int msgid, int cmd,
            struct msgid_ds* buf);
```

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

успех - 0

неуспех - -1 и errno установлена

struct msgid_ds

```
struct msqid_ds {
    struct ipc_perm msg_perm; /* operation permission */
    struct msg*msg_first; /* ptr to first message on q */
    struct msg*msg_last; /* ptr to last message on q */
    ulong msg_cbytes; /* current # bytes on q */
    ulong msg_qnum; /* # of messages on q */
    ulong msg_qbytes; /* max # of bytes on q */
    pid_t msg_lspid; /* pid of last msgsnd */
    pid_t msg_lrpid; /* pid of last msgrcv */
    time_t msg_stime; /* last msgsnd time */
    long msg_stimfrac; /* reserved for time_t expansion */
    time_t msg_rtime; /* last msgrcv time */
    long msg_rtimfrac;
    time_t msg_ctime; /* last change time */
    long msg_ctimfrac;
};
```

struct msgbuf

```
struct msgbuf {  
    long mtype; /* message type */  
    char mtext[1]; /*message text*/  
};
```

msgop(2)

ИСПОЛЬЗОВАНИЕ

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
int msgsnd (int msg id, const struct msgbuf *msgp, int
    msgsz, int msgflg);
int msgrcv (int msg id, const struct msgbuf *msgp, int
    msgsz, long msgtyp, int msgflg);
```

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

msgsnd успех - 0

msgrcv успех - количество прочитанных байтов

неуспех - -1 и errno установлена

СЕМАФОРЫ

- Разделяемое короткое беззнаковое целое значение
- Используется для:
 - блокировки
 - управления доступом к ресурсам
 - подсчета
 - разделения небольшого беззнакового значения между процессами

semget(2)

ИСПОЛЬЗОВАНИЕ

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

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

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

```
int semget (key_t key, int nitems, int  
    semfl);
```

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

успех - неотрицательный идентификатор семафора

неуспех - -1 и errno установлена

semctl(2)

ИСПОЛЬЗОВАНИЕ

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
int semctl (int semid,
            int semnum, int cmd,
            union semun arg);
```

union semun

```
union semun {  
    int val;  
    struct semid_ds *buf;  
    ushort *array;  
};
```

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

GETVAL - значение семафора

GETPID - идентификатор процесса, совершившего последнюю операцию над семафором

GETNCNT - количество процессов, ожидающих увеличения значения семафора по сравнению с текущим значением

GETZCNT - количество процессов, ожидающих нулевого значения семафора

неуспех - -1 и errno установлена

semop(2)

ИСПОЛЬЗОВАНИЕ

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
int semop(int semid,
    struct sembuf *sops, unsigned nsops);
```

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

успех - ноль

неуспех - -1 и errno установлена

struct sembuf

```
struct sembuf {  
    ushort sem_num; /* semaphore */  
    short sem_op;  
    /* semaphore operation */  
    short sem_flg;  
    /* operation flags */  
};
```

Флаги

- `IPC_NOWAIT` – неблокирующая операция
- `SEM_UNDO` – отмена операции над семафором при завершении программы

РАЗДЕЛЯЕМАЯ ПАМЯТЬ

- `shmget` - создать или получить доступ к сегменту разделяемой памяти
- `shmctl` - определить состояние разделяемого сегмента; изменить хозяина/группу сегмента и права доступа; удалить сегмент
- `shmat` - присоединить (`shmat`) разделяемый сегмент к области данных процесса, или отсоединить его (`shmdt`)

shmget(2)

ИСПОЛЬЗОВАНИЕ

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
int shmget (key_t key, int size,
            int shmflg);
```

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

успех - неотрицательный идентификатор
разделяемого сегмента

неуспех - -1 и errno установлена

shmctl(2)

ИСПОЛЬЗОВАНИЕ

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

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

```
#include <sys/shm.h>
```

```
int shmctl (int shmid, int cmd, struct  
    shm_id *buf);
```

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

успех - 0

неуспех - -1 и errno установлена

shmop(2)

ИСПОЛЬЗОВАНИЕ

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
void *shmat(int shmid, void *shmaddr,
            int shmflg);
int shmdt (void *shmaddr);
```

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

shmat успех - виртуальный адрес начала сегмента

shmdt успех - 0

неуспех - -1 и errno установлена