Next: Работа с сокетами **Up:** Примеры **Previous:** Семафоры Contents Index

Разделяемые сегменты памяти

```
/* Программа иллюстрирует
   возможности системного вызова shmctl()
   (операции управления разделяемыми сегментами) */

#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

main ()
{
  extern int errno;
  int rtrn, shmid, command, choice;
  struct shmid_ds shmid_ds, *buf;
  buf = &shmid_ds;

  /* Ввести идентификатор сегмента и действие */
  printf ("Введите идентификатор shmid: ");
  scanf ("%d", &shmid);

  printf ("Введите номер требуемого действия:\n");
  printf ("  IPC_STAT   = 1\n");
  printf ("  IPC_SET    = 2\n");
```

```
        printf ("  IPC_RMID   = 3\n");
        printf ("  SHM_LOCK   = 4\n");
        printf ("  SHM_UNLOCK = 5\n");
        printf ("  Выбор      = ");
        scanf ("%d", &command);

        /* Проверить значения */
        printf ("\nидентификатор = %d, действие = %d\n",
                shmid, command);

        switch (command) {
          case 1:     /* Скопировать информацию
                         о состоянии разделяемого сегмента
                         в пользовательскую структуру
                         и вывести ее */

            rtrn = shmctl (shmid, IPC_STAT, buf);
            printf ("\nИд-р пользователя = %d\n",
                    buf->shm_perm.uid);
            printf ("Ид-р группы пользователя = %d\n",
                    buf->shm_perm.gid);
            printf ("Ид-р создателя = %d\n",
                    buf->shm_perm.cuid);
            printf ("Ид-р группы создателя = %d\n",
                    buf->shm_perm.cgid);
            printf ("Права на операции = 0%o\n",
                    buf->shm_perm.mode);
            printf ("Последовательность номеров ");
                    buf->shm_perm.cgid);
            printf ("используемых слотов = 0%x\n",
                    buf->shm_perm.seq);
            printf ("Ключ = 0%x\n", buf->shm_perm.key);
            printf ("Размер сегмента = %d\n", buf->shm_segsz);
            printf ("Выполнил последнюю операцию = %d\n",
                    buf->shm_lpid);
            printf ("Создал сегмент = %d\n", buf->shm_cpid);
            printf ("Число присоединивших сегмент = %d\n",
```

```
                buf->shm_nattch);
        printf ("Число удерживающих в памяти = %d\n",
                buf->shm_cnattch);
        printf ("Последнее присоединение = %d\n",
                buf->shm_atime);
        printf ("Последнее отсоединение = %d\n",
                 buf->shm_dtime);
        printf ("Последнее изменение = %d\n",
                 buf->shm_ctime);
        break;

      case 2:     /* Выбрать и изменить поле (поля)
                     ассоциированной структуры данных */
        /* Получить исходные значения структуры данных */
        rtrn = shmctl (shmid, IPC_STAT, buf);
        printf ("Введите номер изменяемого поля:\n");
        printf ("  shm_perm.uid  = 1\n");
        printf ("  shm_perm.gid  = 2\n");
        printf ("  shm_perm.mode = 3\n");
        printf ("  Выбор         = ");
        scanf ("%d", &choice);

        switch (choice) {
          case 1:
            printf ("\nВведите ид-р пользователя: "),
            scanf ("%d", &buf->shm_perm.uid);
            printf ("\nИд-р пользователя = %d\n",
                    buf->shm_perm.uid);
            break;

          case 2:
            printf ("\nВведите ид-р группы: "),
            scanf ("%d", &buf->shm_perm.gid);
            printf ("\nИд-р группы = %d\n",
                    buf->shm_perm.uid);
            break;
```

```
            case 3:
              printf ("\nВведите восьмеричный код прав: ");
              scanf ("%o", &buf->shm_perm.mode);
              printf ("\nПрава на операции = 0%o\n",
                      buf->shm_perm.mode);
              break;
           }

          /* Внести изменения */
          rtrn = shmctl (shmid, IPC_SET, buf);
          break;

        case 3:      /* Удалить идентификатор и
                        ассоциированную структуру данных */
          rtrn = shmctl (shmid, IPC_RMID, NULL);
          break;


        case 4:     /* Удерживать разделяемый сегмент
                       в памяти */
          rtrn = shmctl (shmid, SHM_LOCK, NULL);
          break;

        case 5:    /* Перестать удерживать сегмент в памяти */
          rtrn = shmctl (shmid, SHM_UNLOCK, NULL);
      }

      if (rtrn == -1) {
        /* Сообщить о неудачном завершении */
        printf ("\nshmctl завершился неудачей!\n");
        printf ("\nКод ошибки = %d\n", errno);
      }
      else {
        /* При успешном завершении сообщить ид-р shmid */
        printf ("\nshmctl завершился успешно, ");
        printf ("идентификатор shmid = %d\n", shmid);
      }
```

```
  exit (0);
}



/* Программа иллюстрирует
   возможности системных вызовов shmat() и shmdt()
   (операции над разделяемыми сегментами памяти) */
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

main ()
{
  extern int errno;
  int shmid, shmaddr, shmflg;
  int flags, attach, detach, rtrn, i;

  /* Цикл присоединений для данного процесса */
  printf ("\nВведите число присоединений ");
  printf ("для процесса (1-4): ");
  scanf ("%d", &attach);
  printf ("\nЧисло присоединений = %d\n", attach);

  for (i = 0; i < attach; i++) {
    /* Ввести идентификатор разделяемого сегмента */
    printf ("\nВведите ид-р разделяемого сегмента,\n");
    printf ("над которым нужно выполнить операции: ");
    scanf ("%d", &shmid);
    printf ("\nИд-р сегмента = %d\n", shmid);

    /* Ввести адрес присоединения */
    printf ("\nВведите адрес присоединения ");
    printf ("в шестнадцатеричной записи: ");
    scanf ("%x", &shmaddr);
```

```
        printf ("\nАдрес присоединения = 0x%x\n", shmaddr);

      /* Выбрать требуемые флаги */
      printf ("\nВведите номер нужной комбинации флагов:\n");
      printf ("  SHM_RND              = 1\n");
      printf ("  SHM_RDONLY           = 2\n");
      printf ("  SHM_RND и SHM_RDONLY = 3\n");
      printf ("  Выбор                = ");
      scanf ("%d", &flags);

      switch (flags) {
        case 1:
          shmflg = SHM_RND;
          break;
        case 2:
          shmflg = SHM_RDONLY;
          break;
        case 3:
          shmflg = SHM_RND | SHM_RDONLY;
          break;
      }
      printf ("\nФлаги = 0%o", shmflg);

      /* Выполнить системный вызов shmat */
      rtrn = shmat (shmid, shmaddr, shmflg);
      if (rtrn == -1) {
        printf ("\nshmat завершился неудачей!\n");
        printf ("\Код ошибки = %d\n", errno);
      }
      else {
        printf ("\nshmat завершился успешно.\n");
        printf ("Идентификатор shmid = %d\n", shmid);
        printf ("Адрес = 0x%x\n", rtrn);
      }
    }

    /* Цикл отсоединений для данного процесса */
```

```
        printf ("\nВведите число отсоединений ");
        printf ("для процесса (1-4): ");
        scanf ("%d", &detach);
        printf ("\nЧисло отсоединений = %d\n", detach);

        for (i = 0; i < detach; i++) {
          /* Ввести адрес отсоединения */
          printf ("\nВведите адрес отсоединяемого сегмента ");
          printf ("в шестнадцатеричной записи: ");
          scanf ("%x", &shmaddr);
          printf ("\nАдрес отсоединения = 0x%x\n", shmaddr);

          /* Выполнить системный вызов shmdt */
          rtrn = shmdt (shmaddr);
          if (rtrn == -1) {
            printf ("\nshmdt завершился неудачей!\n");
            printf ("\Код ошибки = %d\n", errno);
          }
          else {
            printf ("\nshmdt завершился успешно,\n");
            printf ("идентификатор shmid = %d\n", shmid);
          }
        }

        exit (0);
      }
```

Alex Otwagin 2002-12-16