Next: Разделяемые сегменты памяти **Up:** Примеры **Previous:** Очереди сообщений **Contents** **Index**

Семафоры

```
/* Программа иллюстрирует
   возможности системного вызова semctl()
   (управление семафорами) */

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

#define MAXSETSIZE 25

main ()
{
  extern int errno;
  struct semid_ds semid_ds;
  int length, rtrn, i, c;
  int semid, semnum, cmd, choice;
  union semun {
    int val;
    struct semid_ds *buf;
    ushort array [MAXSETSIZE];
  } arg;
```

```
/* Инициализация указателя на структуру данных */
arg.buf = &semid_ds;

/* Ввести идентификатор множества семафоров */
printf ("Введите ид-р множества семафоров: ");
scanf ("%d", &semid);

/* Выбрать требуемое управляющее действие */
printf ("\nВведите номер требуемого действия:\n");
printf ("  GETVAL   = 1\n");
printf ("  SETVAL   = 2\n");
printf ("  GETPID   = 3\n");
printf ("  GETNCNT  = 4\n");
printf ("  GETZCNT  = 5\n");
printf ("  GETALL   = 6\n");
printf ("  SETALL   = 7\n");
printf ("  IPC_STAT = 8\n");
printf ("  IPC_SET  = 9\n");
printf ("  IPC_RMID = 10\n");
printf ("  Выбор    = ");
scanf ("%d", &cmd);

/* Проверить значения */
printf ("идентификатор = %d, команда = %d\n",
        semid, cmd);

/* Сформировать аргументы и выполнить вызов */
switch (cmd) {
  case 1:     /* Получить значение */
    printf ("\nВведите номер семафора: ");
    scanf ("%d", &semnum);
    /* Выполнить системный вызов */
    rtrn = semctl (semid, semnum, GETVAL, 0);
    printf ("\n3начение семафора = %d\n", rtrn);
    break;

  case 2:     /* Установить значение */
```

```
  printf ("\nВведите номер семафора: ");
  scanf ("%d", &semnum);
  printf ("\nВведите значение: ");
  scanf ("%d", &arg.val);
  /* Выполнить системный вызов */
  rtrn = semctl (semid, semnum, SETVAL, arg.val);
  break;

case 3:     /* Получить ид-р процесса */
 rtrn = semctl (semid, 0, GETPID, 0);
 printf ("\Последнюю операцию выполнил: %d\n",rtrn);
 break;

case 4:    /* Получить число процессов, ожидающих
              увеличения значения семафора */
  printf ("\nВведите номер семафора: ");
  scanf ("%d", &semnum);
  /* Выполнить системный вызов */
  rtrn = semctl (semid, semnum, GETNCNT, 0);
  printf ("\nЧисло процессов = %d\n", rtrn);
  break;

case 5:    /* Получить число процессов, ожидающих
              обнуления значения семафора */
  printf ("Введите номер семафора: ");
  scanf ("%d", &semnum);
  /* Выполнить системный вызов */
  rtrn = semctl (semid, semnum, GETZCNT, 0
  printf ("\nЧисло процессов = %d\n", rtrn);
  break;

case 6:    /* Опросить все семафоры */
  /* Определить число семафоров в множестве */
  rtrn = semctl (semid, 0, IPC_STAT, arg.buf);
  length = arg.buf->sem_nsems;
  if (rtrn == -1)
    goto ERROR;
```

```
      /* Получить и вывести значения всех
         семафоров в указанном множестве */
      rtrn = semctl (semid, 0, GETALL, arg.array);
      for (i = 0; i < length; i++)
        printf (" %d", arg.array [i]);
      break;

    case 7:    /* Установить все семафоры */
      /* Определить число семафоров в множестве */
      rtrn = semctl (semid, 0, IPC_STAT, arg.buf);
      length = arg.buf->sem_nsems;
      if (rtrn == -1)
        goto ERROR;
      printf ("\nЧисло семафоров = %d\n", length);
      /* Установить значения семафоров множества */
      printf ("\nВведите значения:\n");
      for (i = 0; i < length; i++)
        scanf ("%d", &arg.array [i]);
      /* Выполнить системный вызов */
      rtrn = semctl (semid, 0, SETALL, arg.array);
      break;

    case 8:     /* Опросить состояние множества */
      rtrn = semctl (semid, 0, IPC_STAT, arg.buf);
      printf ("\nИдентификатор пользователя = %d\n",
              arg.buf->sem_perm.uid
      printf ("Идентификатор группы = %d\n",
              arg.buf->sem_perm.gid);
      printf ("Права на операции = 0%o\n",
              arg.buf->sem_perm.mode);
      printf ("Число семафоров в множестве = %d\n",
              arg.buf->sem_nsems);
      printf ("Время последней операции = %d\n",
              arg.buf->sem_otime);
      printf ("Время последнего изменения = %d\n",
              arg.buf->sem_ctime);
      break;
```

```
case 9:    /* Выбрать и изменить поле
              ассоциированной структуры данных */
  /* Опросить текущее состояние */
  rtrn = semctl (semid, 0, IPC_STAT, arg.buf);
  if (rtrn == -1)
    goto ERROR;

  printf ("\nВведите номер поля, ");
  printf ("которое нужно изменить: \n");
  printf ("  sem_perm.uid  = 1\n");
  printf ("  sem_perm.gid  = 2\n");
  printf ("  sem_perm.mode = 3\n");
  printf ("  Выбор         = ");
  scanf ("%d", &choice);

  switch (choice) {
    case 1:         /* Изменить ид-р владельца */
      printf ("\nВведите ид-р владельца: ");
      scanf ("%d", &arg.buf->sem_perm.uid);
      printf ("\nИд-р владельца = %d\n",
              arg.buf->sem_perm.uid);
      break;

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

    case 3:      /* Изменить права на операции */
      printf ("\nВведите восьмеричный код прав: ");
      scanf ("%o", &arg.buf->sem_perm.mode);
      printf ("\nПрава = 0%o\n",
              arg.buf->sem_perm.mode);
      break;
```

```
        }

        /* Внести изменения */
        rtrn = semctl (semid, 0, IPC_SET, arg.buf);
        break;

      case 10:    /* Удалить ид-р множества семафоров и
                     ассоциированную структуру данных */
        rtrn = semctl (semid, 0, IPC_RMID, 0);
    }
    if (rtrn == -1) {
      /* Сообщить о неудачном завершении */
      ERROR:
      printf ("\nsemctl завершился неудачей!\n");
      printf ("\nКод ошибки = %d\n", errno);
    }
    else {
      printf ("\nmsgctl завершился успешно,\n");
      printf ("идентификатор semid = %d\n", semid);
    }

    exit (0);
  }



/* Программа иллюстрирует
   возможности системного вызова semop()
   (операции над множеством семафоров) */


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

#define MAXOPSIZE 10
```

```
main ()
{
  extern int errno;
  struct sembuf sops [MAXOPSIZE];
  int semid, flags, i, rtrn;
  unsigned nsops;

/* Ввести идентификатор множества семафоров */
printf ("\nВведите идентификатор множества семафоров,");
printf ("\nнад которым будут выполняться операции: ");
scanf ("%d", &semid);
printf ("\nИд-р множества семафоров = %d", semid);

/* Ввести число операций */
printf ("\nВведите число операций ");
printf ("над семафорами из этого множества: \n");
scanf ("%d", &nsops);
printf ("\nЧисло операций = %d", nsops);

/* Инициализировать массив операций */
for (i = 0; i < nsops; i++) {
  /* Выбрать семафор из множества */
  printf ("\nВведите номер семафора: ");
  scanf ("%d", &sops [i].sem_num);
  printf ("\nНомер = %d", sops [i].sem_num);

/* Ввести число, задающее операцию */
printf ("\nЗадайте операцию над семафором: ");
scanf ("%d", &sops [i].sem_op);
printf ("\nОперация = %d", sops [i].sem_op);

/* Указать требуемые флаги */
printf ("\nВведите код, ");
printf ("соответствующий требуемым флагам:\n");
printf ("  Нет флагов            = 0\n");
printf ("  IPC_NOWAIT            = 1\n");
```

```
        printf ("  SEM_UNDO              = 2\n");
        printf ("  IPC_NOWAIT и SEM_UNDO = 3\n");
        printf ("  Выбор                 = ");
        scanf ("%d", &flags);

        switch (flags) {
          case 0:
            sops [i].sem_flg = 0;
            break;
          case 1:
            sops [i].sem_flg = IPC_NOWAIT;
            break;
          case 2:
            sops [i].sem_flg = SEM_UNDO;
            break;
          case 3:
            sops [i].sem_flg = IPC_NOWAIT | SEM_UNDO;
            break;
        }
        printf ("\nФлаги = 0%o", sops [i].sem_flg);
      }

      /* Распечатать все структуры массива */
      printf ("\nМассив операций:\n");
      for (i = 0; i < nsops; i++) {
        printf ("  Номер семафора = %d\n", sops [i].sem_num);
        printf ("  Операция = %d\n", sops [i].sem_op);
        printf ("  Флаги = 0%o\n", sops [i].sem_flg);
      }

      /* Выполнить системный вызов */
      rtrn = semop (semid, sops, nsops);
      if (rtrn == -1) {
        printf ("\nsemop завершился неудачей!\n");
        printf ("Код ошибки = %d\n", errno);
      }
      else {
```

```
        printf ("\nsemop завершился успешно.\n");
        printf ("Идентификатор semid = %d\n", semid);
        printf ("Возвращенное значение = %d\n", rtrn);
      }

      exit (0);
    }
```

Alex Otwagin 2002-12-16