Next: Семафоры Up: Примеры Previous: Примеры Contents Index

Очереди сообщений

```
/* Программа иллюстрирует
   возможности системного вызова msgget()
   (получение идентификатора очереди сообщений) */

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

main ()
{
  key_t key;        /* Тип описан как целое */
  int opperm, flags;    /* Права на операции и флаги */
  int msgflg, msqid;

  /* Ввести требуемый ключ */
  printf ("\nВведите шестнадцатеричный ключ: ");
  scanf ("%x", &key);

  /* Ввести права на операции */
  printf ("\nВведите права на операции ");
  printf ("в восьмеричной записи: ");
```

```
    scanf ("%o", &opperm);

    /* Установить требуемые флаги */
    printf ("\nВведите код, соответствущий ");
    printf ("нужной комбинации флагов:\n");
    printf ("  Нет флагов           = 0\n");
    printf ("  IPC_CREAT            = 1\n");
    printf ("  IPC_EXCL             = 2\n");
    printf ("  IPC_CREAT и IPC_EXCL = 3\n");
    printf ("  Выбор                = ");

    /* Получить флаги, которые нужно установить */
    scanf ("%d", &flags);

    /* Проверить значения */
    printf ("\nключ = 0x%x, права = 0%o, флаги = %d\n",
            key, opperm, flags);

    /* Объединить флаги с правами на операции */
  switch (flags) {
    case 0:     /* Флаги не устанавливать */
      msgflg = (opperm | 0);
      break;
    case 1:     /* Установить флаг IPC_CREAT */
      msgflg = (opperm | IPC_CREAT);
      break;
    case 2:     /* Установить флаг IPC_EXCL */
      msgflg = (opperm | IPC_EXCL);
      break;
    case 3:     /* Установить оба флага */
      msgflg = (opperm | IPC_CREAT | IPC_EXCL);
  }

  /* Выполнить системный вызов msgget */
  msqid = msgget (key, msgflg);

  if (msqid == -1) {
```

```
      /* Сообщить о неудачном завершении */
      printf ("\nmsgget завершился неудачей!\n"
      printf ("Код ошибки = %d\n", errno);
    }
    else
      /* При успешном завершении сообщить msqid */
      printf ("\nИдентификатор msqid = %d\n", msqid);

    exit (0);
    }



  /* Программа иллюстрирует
     возможности системного вызова msgctl()
     (управление очередями сообщений) */

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

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

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

    printf ("Введите номер требуемого действия:\n");
    printf ("  IPC_STAT = 1\n");
    printf ("  IPC_SET  = 2\n");
    printf ("  IPC_RMID = 3\n");
```

```
    printf ("  Выбор    = ");
    scanf ("%d", &command);

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

    switch (command) {
      case 1: /* Скопировать информацию
                 о состоянии очереди сообщений
                 в пользовательскую структуру
                 и вывести ее */
        rtrn = msgctl (msqid, IPC_STAT, buf);
        printf ("\n Идентификатор пользователя = %d\n",
                buf->msg_perm.uid);
        printf ("\n Идентификатор группы = %d\n",
                buf->msg_perm.gid);
        printf ("\n Права на операции = 0%o\n",
                buf->msg_perm.mode);
        printf ("\n Размер очереди в байтах = %d\n",
                buf->msg_qbytes);
        break;

      case 2: /* Выбрать и изменить поле (поля)
                  ассоциированной структуры данных */
        /* Сначала получить исходное значение
           структуры данных */
        rtrn = msgctl (msqid, IPC_STAT, buf);
        printf ("\nВведите номер поля, ");
        printf ("которое нужно изменить:\n");
        printf ("  msg_perm.uid  = 1\n");
        printf ("  msg_perm.gid  = 2\n");
        printf ("  msg_perm.mode = 3\n");
        printf ("  msg_qbytes    = 4\n");
        printf ("  Выбор         = ");
        scanf ("%d", &choice);
```

```
        switch (choice) {
          case 1:
            printf ("\nВведите ид-р пользователя: ");
            scanf ("%d", &buf->msg_perm.uid);
            printf ("\nИд-р пользователя = %d\n",
                    buf->msg_perm.uid);
            break;
          case 2:
            printf ("\nВведите ид-р группы: ");
            scanf ("%d", &buf->msg_perm.gid);
            printf ("\nИд-р группы = %d\n",
                    buf->msg_perm.uid);
            break;

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

          case 4:
            printf ("\nВведите размер очереди = ");
            scanf ("%d", &buf->msg_qbytes);
            printf ("\nЧисло байт в очереди = %d\n",
                    buf->msg_qbytes);
            break;
        }

        /* Внести изменения */
        rtrn = msgctl (msqid, IPC_SET, buf);
        break;

      case 3: /* Удалить идентификатор и
                 ассоциированные с ним очередь
                 сообщений и структуру данных */
        rtrn = msgctl (msqid, IPC_RMID, NULL);
```

```
    }

    if (rtrn == -1) {
      /* Сообщить о неудачном завершении */
      printf ("\nmsgctl завершился неудачей!\n");
      printf ("\nКод ошибки = %d\n", errno);
    }

    else {
      /* При успешном завершении сообщить msqid */
      printf ("\nmsgctl завершился успешно,\n");
      printf ("идентификатор = %d\n", msqid);
    }

    exit (0);
  }



  /* Программа иллюстрирует
     возможности системных вызовов msgsnd() и msgrcv()
     (операции над очередями сообщений) */

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

  #define MAXTEXTSIZE 8192

  struct msgbufl {
    long mtype;
    char mtext [MAXTEXTSIZE];
  } sndbuf, rcvbuf, *msgp;

  main ()
  {
    extern int errno;
```

```
int flag, flags, choice, rtrn, i, c;
int rtrn, msqid, msgsz, msgflg;
long msgtyp;
struct msqid_ds msqid_ds, *buf;
buf = &msqid_ds;

/* Выбрать требуемую операцию */
printf ("\nВведите код, соответствующий ");
printf ("посылке или приему сообщения:\n");
printf ("  Послать = 1\n");
printf ("  Принять = 2\n");
printf ("  Выбор   = ");
scanf ("%d", &choice);

if (choice == 1) {
  /* Послать сообщение */
  msgp = &sndbuf; /* Указатель на структуру */

  printf ("\nВведите идентификатор ");
  printf ("очереди сообщений,\n");
  printf ("в которую посылается сообщение: ");
  scanf ("%d", &msqid);

  /* Установить тип сообщения */
  printf ("\nВведите положительное число - ");
  printf ("тип сообщения: ");
  scanf ("%d", &msgp->mtype);

  /* Ввести посылаемое сообщение */
  printf ("\nВведите сообщение: \n");

  /* Управляющая последовательность CTRL+D
     завершает ввод сообщения */

  /* Прочитать символы сообщения
     и поместить их в массив mtext */
  for (i = 0; ((c = getchar ()) != EOF); i++)
```

```
    sndbuf.mtext [i] = c;

  /* Определить размер сообщения */
  msgsz = i + 1;

  /* Выдать текст посылаемого сообщения */
  for (i = 0; i < msgsz; i++)
    putchar (sndbuf.mtext [i]);

  /* Установить флаг IPC_NOWAIT, если это нужно */
  printf ("\nВведите 1, если хотите установить ");
  printf ("флаг IPC_NOWAIT: ");
  scanf ("%d", &flag);
  if (flag == 1)
    msgflg = IPC_NOWAIT;
  else
    msgflg = 0;

  /* Проверить флаг */
  printf ("\nФлаг = 0%o\n", msgflg);

  /* Послать сообщение */
  rtrn = msgsnd (msqid, msgp, msgsz, msgflg);
  if (rtrn == -1) {
    printf ("\nmsgsnd завершился неудачей!\n");
    printf ("Код ошибки = %d\n", errno);
  }
  else {
    /* Вывести результат; при успешном
       завершении он должен равняться нулю */
    printf ("\nРезультат = %d\n", rtrn);

    /* Вывести размер сообщения */
    printf ("\nРазмер сообщения = %d\n", msgsz);

    /* Опрос измененной структуры данных */
    msgctl (msqid, IPC_STAT, buf);
```

```
        /* Вывести изменившиеся поля */
        printf ("Число сообщений в очереди = %d\n",
                buf->msg_qnum);
        printf ("Ид-р последнего отправителя = %d\n",
                buf->msg_lspid);
        printf ("Время последнего отправления = %d\n",
                buf->msg_stime);
      }
    }

    if (choice == 2) {
      /* Принять сообщение */
      msgp = &rcvbuf;

      /* Определить нужную очередь сообщений */
      printf ("\nВведите ид-р очереди сообщений: ");
      scanf ("%d", &msqid);

      /* Определить тип сообщения */
      printf ("\nВведите тип сообщения: ");
      scanf ("%d", &msgtyp);

      /* Сформировать управляющие флаги
         для требуемых действий */
      printf ("\nВведите код, соответствущий ");
      printf ("нужной комбинации флагов:\n");
      printf ("  Нет флагов               = 0\n");
      printf ("  MSG_NOERROR              = 1\n");
      printf ("  IPC_NOWAIT               = 2\n");
      printf ("  MSG_NOERROR и IPC_NOWAIT = 3\n");
      printf ("  Выбор                    = ");
      scanf ("%d", &flags);

      switch (flags) {
        /* Установить msgflg как побитное ИЛИ
           соответствующих констант */
```

```
          case 0:
            msgflg = 0;
            break;
          case 1:
            msgflg = MSG_NOERROR;
            break;
          case 2:
            msgflg = IPC_NOWAIT;
            break;
          case 3:
            msgflg = MSG_NOERROR | IPC_NOWAIT;
            break;
        }

        /* Определить, какое число байт принять */
        printf ("\nВведите число байт, которое ");
        printf ("нужно принять (msgsz): ");
        scanf ("%d", &msgsz);

        /* Проверить значение аргументов */
        printf ("\nИдентификатор msqid = %d\n", msqid);
        printf ("Тип сообщения = %d\n", msgtyp);
        printf ("Число байт = %d\n", msgsz);
        printf ("Флаги = %o\n", msgflg);

        /* Вызвать msgrcv для приема сообщения */
        rtrn = msgrcv (msqid, msgp, msgsz, msgtyp, msgflg);

        if (rtrn == -1) {
          printf ("\nmsgrcv завершился неудачей!\n");
          printf ("Код ошибки = %d\n", errno);
        }
        else {
          printf ("\nmsgrcv завершился успешно,\n");
          printf ("идентификатор очереди = %d\n", msqid);

            /* Напечатать число принятых байт,
```

```
            оно равно возвращаемому значению */
          printf ("Принято байт: %d\n", rtrn);

          /* Распечатать принятое сообщение */
          for (i = 0; i < rtrn; i++)
            putchar (rcvbuf.mtext [i]);
      }

      /* Опрос ассоциированной структуры данных */
      msgctl (msqid, IPC_STAT, buf);

      printf ("\nЧисло сообщений в очереди = %d\n",
              buf->msg_qnum);
      printf ("Ид-р последнего получателя = %d\n",
              buf->msg_lrpid);
      printf ("Время последнего получения = %d\n",
              buf->msg_rtime);
    }

    exit (0);
  }
```

Alex Otwagin 2002-12-16