Министерство образования и науки Российской Федерации

Новосибирский государственный технический университет

Кафедра программных систем и баз данных

Лабораторная работа №7 по дисциплине

«Управление ресурсами в вычислительных системах»

Факультет: ПМИ

Группа: ПМ-04

Студентки: Анисимова К.Н.

Сулейманова К.А.

Вариант: 4

Преподаватель: Быханов К.В.

Куликов И.М.

Новосибирск

2013

1. ***Цель работы***

Освоение средств IPC. Написание программ, использующих механизм семафоров, очередей сообщений, сегментов разделяемой памяти.

1. ***Задание***

Программа моделирует работу примитивной СУБД, хранящей единственную таблицу в оперативной памяти. Выполняя некоторые циклы работ, K порожденных процессов посредством очереди сообщений передают родительскому процессу номер строки, которую нужно удалить из таблицы. Родительский процесс выполняет указанную операцию и возвращает содержимое удалённой строки.

1. ***Решение***

*Алгоритм работы*

Пусть содержание таблицы содержится в файле конфигурации. Тогда после чтения и проверки их правильности основной процесс создает необходимое количество потомков, каждый из которых посылает сообщение в очередь с номер строки, которое он хочет удалить из таблицы (например порядковый номер созданного процесса). Основной процесс получает эти сообщения и по возможности удаляет строки с такими номерами из таблицы.

1. ***Средства программирования***

В данной работе были использованы *функции:*

* Системный вызов для создания очереди сообщений

int msgqid = msgget(key\_t key, int flag).

* Для помещения сообщения в очередь использовали системный вызов msgsnd():

int msgsnd (int msgqid, void \*msg, size\_t size, int flag),

* Для приема сообщения использовали системный вызов msgrcv():

nt msgrcv (int msgqid, void \*msg, size\_t size, long msg\_type, int flag);

* Системный вызов msgctl()

int msgctl (int msgqid, int command, struct msqid\_ds \*msg\_stat)

используется

* для опроса состояния описателя очереди сообщений (command = IPC\_STAT) и помещения его в структуру msg\_stat (детали опускаем);
* изменения его состояния (command = IPC\_SET), например, изменения прав доступа к очереди;
* для уничтожения указанной очереди сообщений (command = IPC\_RMID).

1. ***Текст программы***

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <stdio.h>

#include <signal.h>

#include <string.h>

#define K 5 //кол-во записей в таблице и кол-во потомков

struct table //таблица

{

char msg[20];

} ;

struct message // структура сообщения

{

long mtype; /\* тип сообщения \*/

char mtext[20]; /\* текст сообщения (SOMEVALUE - любое) \*/

};

void read(struct table tab[K], FILE \*f)//ввод таблицы

{

int i;

char s[20];

for (i=0; i<K; i++)

{

fscanf(f, "%s", &s);

strcpy(tab[i].msg, s);;

}

}

void add\_queue(int i, int done) //добавление сообщение в очередь сообщений

{

struct message msg;

if (done < 0)

printf("ERROR KEY");

else

{

//формируем сообщение

sprintf(msg.mtext,"%d %d",getpid(),i);//добавляем в сообщение идентификатор потомка и номер строки

msg.mtype=1;//тип = 1

//добавляем в очередь

msgsnd(done,(void\*)&msg,20,0);

}

}

void delete\_tab(struct table tab[5], int msgid\_in) //удаление строк из таблицы

{

struct message msg;

char res[20];

int i;

int row,pid;

//ждем по 1му сообщению от каждого потомка

for(i=0; i<K; i++ )

{

//получаем сообщение

printf("\nEEE!\n");

msgrcv(msgid\_in,&msg,20,1,0);

//считываем номер строки

sscanf(msg.mtext,"%d %d",&pid,&row);

//удаляем

strcpy(res, tab[row].msg);

printf("proc %d delete %s", pid, res);

tab[row].msg[0] = '\0';

}

}

int main()

{

struct table tab[K];

int i , j, pid\_pr;

int pids[K];

int msg\_in;

key\_t qkey\_in= IPC\_PRIVATE; //ключ для очереди

//открываем файл с данными для таблицы

FILE \*f\_in = fopen("data.txt", "r");

read(tab,f\_in);

//создаем очередь

msg\_in = msgget(qkey\_in, 0666 | IPC\_CREAT);

if( msg\_in < 0 )

{

printf("ERROR : queue don't create\n");

return 1;

}

//создаем потомков

for(i=0; i<K; i++)

{

switch(pid\_pr = fork())

{

case -1:

{

printf("ERROR");

for(j=0; j<i; j++)

kill(j, SIGKILL);

}

case 0:

{

add\_queue(i,msg\_in);

return 0;

}

default:

{

pids[i] = pid\_pr;

}

}

}

delete\_tab(tab, msg\_in);

msgctl(msg\_in,IPC\_RMID,NULL);

return 0;

}

1. ***Тесты***
2. data.txt:

1

2

3

4

5

Результат:

proc 3904 delete 1

proc 3905 delete 2

proc 3906 delete 3

proc 3907 delete 4

proc 3908 delete 5

1. data.txt:

СООБЩЕНИЕ1

СООБЩЕНИЕ2

СООБЩЕНИЕ3

СООБЩЕНИЕ4

СООБЩЕНИЕ5

Результат:

proc 3965 delete СООБЩЕНИЕ1

proc 3969 delete СООБЩЕНИЕ5

proc 3968 delete СООБЩЕНИЕ4

proc 3967 delete СООБЩЕНИЕ3

proc 3966 delete СООБЩЕНИЕ2

1. data.txt:

СООБЩЕНИЕ1

СООБЩЕНИЕ2

СООБЩЕНИЕ3

СООБЩЕНИЕ4

СООБЩЕНИЕ5

СООБЩЕНИЕ6

СООБЩЕНИЕ7

СООБЩЕНИЕ8

Результат:

proc 4034 delete СООБЩЕНИЕ2

proc 4035 delete СООБЩЕНИЕ3

proc 4037 delete СООБЩЕНИЕ5

proc 4036 delete СООБЩЕНИЕ4

proc 4038 delete СООБЩЕНИЕ6

proc 4033 delete СООБЩЕНИЕ1

proc 4039 delete СООБЩЕНИЕ7

proc 4040 delete СООБЩЕНИЕ8