Министерство образования и науки Российской Федерации

Новосибирский государственный технический университет

Кафедра теоретической и прикладной информатики

Лабораторная работа №5 по дисциплине

«Управление ресурсами в вычислительных системах»

Факультет: ПМИ

Группа: ПМИ-12

Студенты: Швадченко А. В.

Михайловский М. А.

Вариант: 4

Преподаватели: Стасышин В. М., Сивак М. А.

Новосибирск

2024

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. ***Описание используемых структур***

• 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():

int 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

o для опроса состояния описателя очереди сообщений (command = IPC\_STAT) и помещения его в структуру msg\_stat;

o изменения его состояния (command = IPC\_SET), например, изменения прав доступа к очереди;

o для уничтожения указанной очереди сообщений (command = IPC\_RMID).

1. ***Спецификация***

Программа находится в каталоге /root/5lr и называется prog.c.

Чтобы получить исполняемый файл необходимо выполнить команду: gcc –o [имя\_исполняемого\_файла] prog.c. Далее можно запустить исполняемый файл командой: ./[ имя\_исполняемого\_файла] [количество строк].

1. ***Тесты***

data.txt:

mes1

mes2

mes3

mes4

mes5

mes6

./proc 3

Результат:

proc1 5411 delete mes1

proc1 5413 delete mes3

proc1 5412 delete mes2`

1. ***Текст программы***

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <stdio.h>

#include <signal.h>

#include <string.h>

#include <time.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, int V)//ввод таблицы

{

int i;

char s[20];

for (i=0; i<V; 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);

}

}

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

{

struct message msg;

char\* res, \*r;

int i;

int row,pid;

//ждем по 1му сообщению от каждого потомка

//получаем сообщени

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

//считываем номер строки

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

//удаляем

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

res = r;

printf("proc1 %d", pid);

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

return res;

}

int main(int argv[], char\*\* argc)

{

struct table tab[K];

int i , j, pid\_pr;

char\* res;

int msg\_in;

int V;

V = atoi(argc[1]);

int pids[V];

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, V);

//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<V; 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;

}

}

}

for(i=1; i<V+1; i++ )

{

res = delete\_tab(tab, msg\_in);

printf(" delete %s\n", res);

}

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

return 0;

}