Министерство образования Республики Беларусь

Учреждение образования

“Брестский Государственный технический университет”

Кафедра ИИТ

Лабораторная работа №6

По дисциплине “Операционные системы”

Тема: “«Средства межпроцессного взаимодействия»”

Выполнил:

Студент 2 курса

Группы ИИ-23

Вышинский А. С.

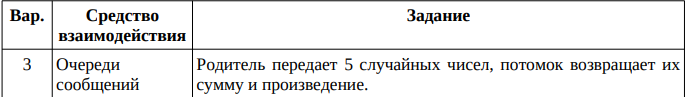
Проверил:

Чеслов А. В.

Брест 2023

**Ход работы:**

Ознакомиться с руководством, теоретическими сведениями и лекционным материалом по использованию и функционированию средств взаимодействия. Написать программу, которая порождает дочерний процесс, и общается с ним через средства взаимодействия согласно варианту (табл.А), передавая и получая информацию согласно варианту (табл.Б). Передачу и получение информации каждым из процессов сопровождать выводом на экран информации типа "процесс такой-то передал/получил такую-то информацию".**Вариант 3.** Написать программу, которая получает со стандартного потока ввода содержимое любого текстового файла и выводит в стандартный поток вывода те строки, в которых больше 3 слов. Протестировать на различных файлах с использованием конвейеров в различных комбинациях вашей программы и команд cat, sort, head, tail.



#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

// Структура для сообщения в очереди

struct MsgBuffer

{

long mtype;

int numbers[5];

};

// Глобальные переменные для хранения идентификатора очереди и PID дочернего процесса

int msgQueueID;

pid\_t childPID;

// Обработчик сигнала SIGUSR1

void sigusr1\_handler(int signo)

{

// Используем только для синхронизации, не требуется никакой обработки

}

// Функция для отправки сообщения в очередь

void send\_message(int numbers[5])

{

struct MsgBuffer msg;

msg.mtype = 1; // Тип сообщения

for (int i = 0; i < 5; ++i)

{

msg.numbers[i] = numbers[i];

}

// Отправляем сообщение

msgsnd(msgQueueID, &msg, sizeof(struct MsgBuffer) - sizeof(long), 0);

printf("Родитель передал числа: %d %d %d %d %d\n", numbers[0], numbers[1], numbers[2], numbers[3], numbers[4]);

}

// Функция для получения сообщения из очереди

void receive\_message()

{

struct MsgBuffer msg;

// Получаем сообщение

msgrcv(msgQueueID, &msg, sizeof(struct MsgBuffer) - sizeof(long), 1, 0);

printf("Потомок получил числа: %d %d %d %d %d\n", msg.numbers[0], msg.numbers[1], msg.numbers[2], msg.numbers[3], msg.numbers[4]);

// Вычисляем сумму и произведение

int sum = 0;

int product = 1;

for (int i = 0; i < 5; ++i)

{

sum += msg.numbers[i];

product \*= msg.numbers[i];

}

// Отправляем результат обратно

msg.mtype = 2; // Тип сообщения для ответа

msg.numbers[0] = sum;

msg.numbers[1] = product;

msgsnd(msgQueueID, &msg, sizeof(struct MsgBuffer) - sizeof(long), 0);

printf("Потомок вернул сумму и произведение: %d %d\n", sum, product);

}

int main()

{

// Создаем очередь сообщений

msgQueueID = msgget(IPC\_PRIVATE, IPC\_CREAT | 0666);

// Устанавливаем обработчик сигнала SIGUSR1

signal(SIGUSR1, sigusr1\_handler);

// Создаем дочерний процесс

childPID = fork();

if (childPID == -1)

{

perror("Ошибка fork");

exit(EXIT\_FAILURE);

}

if (childPID == 0)

{

// Дочерний процесс

pause(); // Ждем сигнала от родителя

// Получаем и обрабатываем сообщение от родителя

receive\_message();

}

else

{

// Родительский процесс

int numbers[5];

// Генерируем случайные числа

for (int i = 0; i < 5; ++i)

{

numbers[i] = rand() % 10 + 1;

}

// Отправляем сигнал SIGUSR1 дочернему процессу

kill(childPID, SIGUSR1);

// Отправляем сообщение с числами дочернему процессу

send\_message(numbers);

// Ждем ответ от дочернего процесса

receive\_message();

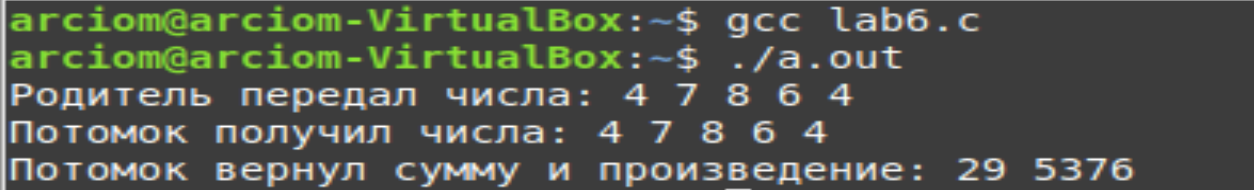
// Удаляем очередь сообщений

msgctl(msgQueueID, IPC\_RMID, NULL);

}

return 0;

}



**Вывод:** Ознакомился с руководством, теоретическими сведениями и лекционным материалом по использованию и функционированию средств взаимодействия. Написал программу, которая порождает дочерний процесс, и общается с ним через средства взаимодействия