Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Курсовой проект по курсу**

**«Операционные системы»**

Студент: Садаков А. А.

Группа: М8О–206Б–19

Вариант: на “удовлетворительно”

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2020.

**Постановка задачи**

## Цель работы

* Приобретение практических навыков в использовании знаний, полученных в течении курса
* Проведение исследование в выбранной предметной области

## Задание

Необходимо спроектировать и реализовать программный прототип в соответствии с выбранным вариантом. Произвести анализ и сделать вывод на основании данных, полученных при работе программного прототипа.

Необходимо написать 3-и программы. Далее будем обозначать эти программы A, B, C. Программа A принимает из стандартного потока ввода строки, а далее их отправляет программе С. Отправка строк должна производится построчно. Программа C печатает в стандартный вывод, полученную строку от программы A. После получения программа C отправляет программе А сообщение о том, что строка получена. До тех, пор пока программа А не примет «сообщение о получение строки» от программы С, она не может отправлять следующую строку программе С. Программа B пишет в стандартный вывод количество отправленных символов программой А и количество принятых символов программой С. Данную информацию программа B получает от программ A и C соответственно. Способ организация межпроцессорного взаимодействия выбирает студент.

**Общие сведения о программе**

В файле **term.h**описана функция **termination**, которая вызывается при получении программой сигналов SIGINT или SIGSEGV. Реализации программ находится в соответствующих .c файлах.

**Основные файлы программы**

**progA.c:**

#include <stdio.h>

#include <string.h>

#include <stdbool.h>

#include "term.h"

#include "zmq.h"

int main () {

signal(SIGINT, termination);

signal(SIGSEGV, termination);

printf("starting A...\n");

bool gotMess = true;

pid\_t myPid = getpid();

void\* contextC = zmq\_ctx\_new();

void\* requestC = zmq\_socket(contextC, ZMQ\_PAIR);

zmq\_connect(requestC, "ipc://connectAC");

void\* contextB = zmq\_ctx\_new();

void\* requestB = zmq\_socket(contextB, ZMQ\_PAIR);

zmq\_connect(requestB, "ipc://connectAB");

zmq\_recv(requestC, &pid[0], sizeof(pid\_t), 0);

zmq\_send(requestC, &myPid, sizeof(pid\_t), 0);

zmq\_recv(requestB, &pid[1], sizeof(pid\_t), 0);

zmq\_send(requestB, &myPid, sizeof(pid\_t), 0);

while (gotMess) {

char line[80] = {'\0'};

printf("enter line: ");

fgets(line, 80, stdin);

size\_t length = strlen(line);

if (length > 0) {

line[length - 1] = '\0';

--length;

}

printf("sending message to C\n");

zmq\_send(requestC, (void\*)line, sizeof(char) \* (80), 0);

zmq\_recv(requestC, &gotMess, sizeof(bool), 0);

zmq\_send(requestB, &length, sizeof(size\_t), 0);

}

zmq\_close(requestC);

zmq\_ctx\_destroy(contextC);

zmq\_close(requestB);

zmq\_ctx\_destroy(contextB);

return 0;

}

**progB.c:**

#include <stdio.h>

#include <string.h>

#include <stdbool.h>

#include "term.h"

#include "zmq.h"

int main () {

signal(SIGINT, termination);

signal(SIGSEGV, termination);

printf("starting B...\n");

bool gotMess = true;

pid\_t myPid = getpid();

void\* contextA = zmq\_ctx\_new();

void\* respondA = zmq\_socket(contextA, ZMQ\_PAIR);

zmq\_bind(respondA, "ipc://connectAB");

void\* contextC = zmq\_ctx\_new();

void\* respondC = zmq\_socket(contextC, ZMQ\_PAIR);

zmq\_bind(respondC, "ipc://connectCB");

zmq\_send(respondA, &myPid, sizeof(pid\_t), 0);

zmq\_recv(respondA, &pid[0], sizeof(pid\_t), 0);

zmq\_recv(respondC, &pid[1], sizeof(pid\_t), 0);

zmq\_send(respondC, &myPid, sizeof(pid\_t), 0);

while (gotMess) {

size\_t length1;

size\_t length2;

zmq\_recv(respondA, &length1, sizeof(size\_t), 0);

zmq\_recv(respondC, &length2, sizeof(size\_t), 0);

printf("send by A: %ld\nreceived by C: %ld\n", length1, length2);

if (length1 == 0) {

gotMess = false;

}

}

zmq\_close(respondA);

zmq\_ctx\_destroy(contextA);

zmq\_close(respondC);

zmq\_ctx\_destroy(contextC);

return 0;

}

**progC.c**

#include <stdio.h>

#include <string.h>

#include <stdbool.h>

#include "term.h"

#include "zmq.h"

int main () {

signal(SIGINT, termination);

signal(SIGSEGV, termination);

printf("starting C...\n");

bool gotMess = true;

pid\_t myPid = getpid();

void\* contextA = zmq\_ctx\_new();

void\* respondA = zmq\_socket(contextA, ZMQ\_PAIR);

zmq\_bind(respondA, "ipc://connectAC");

void\* contextB = zmq\_ctx\_new();

void\* requestB = zmq\_socket(contextB, ZMQ\_PAIR);

zmq\_connect(requestB, "ipc://connectCB");

zmq\_send(respondA, &myPid, sizeof(pid\_t), 0);

zmq\_recv(respondA, &pid[0], sizeof(pid\_t), 0);

zmq\_send(requestB, &myPid, sizeof(pid\_t), 0);

zmq\_recv(requestB, &pid[1], sizeof(pid\_t), 0);

while (gotMess) {

size\_t length;

char line[80];

zmq\_recv(respondA, line, sizeof(char) \* (80), 0);

printf("got message from A: \"%s\"\n", line);

length = strlen(line);

if (length == 0) {

gotMess = false;

}

zmq\_send(respondA, &gotMess, sizeof(bool), 0);

zmq\_send(requestB, &length, sizeof(size\_t), 0);

}

zmq\_close(respondA);

zmq\_ctx\_destroy(contextA);

zmq\_close(requestB);

zmq\_ctx\_destroy(contextB);

return 0;

}

**Пример работы**

progA:

aleksandr201@LAPTOP-M1VJISD6:~/OS/KP$ ./progA

starting A...

enter line: qef fef

sending message to C

enter line: wryyyyyyyyy

sending message to C

enter line: KONO DIO DA

sending message to C

enter line:

sending message to C

progB:

aleksandr201@LAPTOP-M1VJISD6:~/OS/KP$ ./progB

starting B...

send by A: 7

received by C: 7

send by A: 11

received by C: 11

send by A: 11

received by C: 11

send by A: 0

received by C: 0

progC:

aleksandr201@LAPTOP-M1VJISD6:~/OS/KP$ ./progC

starting C...

got message from A: "qef fef"

got message from A: "wryyyyyyyyy"

got message from A: "KONO DIO DA"

got message from A: ""

**Вывод**

Для написания данного курсового проекта был выбран язык C, поскольку задание не сложное и всё можно реализовать средствами этого языка.

Пригодились знания, полученные в ходе работы с ZMQ, а также обработка сигналов (если 1 из программ завершит работу, то остальным будет отправлен сигнал завершения).