Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Курсовая работа по курсу**

**«Операционные системы»**

**Тема работы**

**«Взаимодействие между процессами. Каналы.»**

Студент: Меджидли Исмаил Ибрагим оглы

Группа: М8О-201Б-21

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2023

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

# **Репозиторий**

https://github.com/imedzhidli/Operational-Systems

# **Постановка задачи**

# **Задание**

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.

# **Общие сведения о программе**

CMakeLists.txt - описание процесса сборки проекта  
  
a.cpp - программа А  
  
b.cpp - программа B

c.cpp - программа C

# 

# **Общий метод и алгоритм решения**

Взаимодействие между процессами реализовано с помощью каналов.

Идея решения состоит в следующем: необходимо создать четыре канала для взаимодействия процессов между собой. А именно: первый канал нужен для того, что программа А отправляла строки программе С, второй — для отправки программой А длины строки программе В, третий — для отправки результата программы С программе А, четвёртый — для отправки программой С длину полученной строки программе В.

Программа завершает работу при нажатии клавиш Ctrl + D.

**Исходный код**

**a.c**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <unistd.h>**

**#include <fcntl.h>**

**#include <ctype.h>**

**#include <stdbool.h>**

**#define MIN\_CAP 4**

**#define STDIN 0**

**size\_t read\_string(char \*\*str\_, int fd) { /\*читаем строку из файл. дескрип\*/**

**free(\*str\_);**

**size\_t str\_size = 0;**

**size\_t cap = MIN\_CAP; /\*емкость буфера для чтения строки\*/**

**char \*str = (char\*) malloc(sizeof(char) \* cap); /\*буфер для чтения памяти маллок возвращет указатель на первый байт памяти\*/**

**if (str == NULL) { /\*проверка что маллок успешно все сделал\*/**

**perror("Ошибка выделения памяти");**

**exit(-1);**

**}**

**char c;**

**while (read(fd, &c, sizeof(char)) == 1) {**

**if (c == '\n') { /\*читаем строку из ФД\*/**

**break;**

**}**

**str[(str\_size)++] = c; /\*читаем символ из ФД и добавляем в строку\*/**

**if (str\_size == cap) {**

**str = (char\*) realloc(str, sizeof(char) \* cap \* 3 / 2); /\*если размер == вместимость, то \* 1,5 раз\*/**

**cap = cap \* 3 / 2;**

**if (str == NULL) { /\*если косяк в выделении памяти\*/**

**perror("Ошибка перераспределения памяти");**

**exit(-2);**

**}**

**}**

**}**

**str[str\_size] = '\0';**

**\*str\_ = str;**

**return str\_size; /\*возврат размер считанной строки\*/**

**}**

**size\_t str\_length(char \*str) { /\*считываем длину строки до \0\*/**

**size\_t length = 0;**

**for (int i = 0; str[i] != '\0'; ++i) {**

**++length;**

**}**

**return length;**

**}**

**int main() { /\*создаем 4 канала\*/**

**int ab[2];**

**int ac[2];**

**int ca[2];**

**int cb[2];**

**pipe(ab);**

**pipe(ac);**

**pipe(ca);**

**pipe(cb);**

**int id1 = fork();**

**if (id1 < 0) {**

**perror("Ошибка создания процесса");**

**exit(1);**

**}**

**else if (id1 == 0) { /\*создает дочерний процесс, внутри которого будет запущена**

**программа c.c с аргументами ac[0], ca[1], и cb[1].\*/**

**close(ac[1]); /\*закрываем ненужные дескрипторы\*/**

**close(ca[0]);**

**close(cb[0]);**

**close(ab[0]);**

**close(ab[1]);**

**char pac[3];**

**sprintf(pac, "%d", ac[0]); /\*преобразование целочисленных значений**

**файловых дескрипторов в строки с помощью функции sprintf()\*/**

**char pca[3];**

**sprintf(pca, "%d", ca[1]);**

**char pcb[3];**

**sprintf(pcb, "%d", cb[1]);**

**execl("./c", "./c", pac, pca, pcb, NULL); /\*системный вызова execl(), где первый аргумент - это имя программы,**

**которую необходимо выполнить, второй аргумент - это имя программы, которое будет использоваться**

**в качестве argv[0] в дочернем процессе, остальные аргументы - это аргументы, передаваемые в новую программу.\*/**

**}**

**else {**

**int id2 = fork(); /\*второй процесс для B\*/**

**if (id2 < 0) {**

**perror("Ошибка создания процесса");**

**exit(1);**

**}**

**else if (id2 == 0) {**

**close(ac[0]);**

**close(ac[1]);**

**close(ca[0]);**

**close(ca[1]);**

**close(cb[1]);**

**close(ab[1]);**

**char pcb[2];**

**sprintf(pcb, "%d", ca[0]); /\*аналогично\*/**

**char pab[2];**

**sprintf(pab, "%d", cb[0]);**

**execl("./b", "./b", pcb, pab, NULL);**

**}**

**else {**

**close(ac[0]);**

**close(ca[1]);**

**close(ab[0]);**

**close(cb[1]);**

**close(cb[0]);**

**char \*str = NULL;**

**while ((read\_string(&str, STDIN)) > 0) { /\*читаем строки через read и передаем их через каналы взаимодействия**

**между процессами с помощью write и read\*/**

**size\_t size = str\_length(str);**

**write(ac[1], &size, sizeof(size\_t));**

**write(ac[1], str, size);**

**write(ab[1], &size, sizeof(size\_t));**

**int ok;**

**read(ca[0], &ok, sizeof(ok));**

**}**

**close(ca[0]);**

**close(ac[1]);**

**close(ab[1]);**

**}**

**}**

**return 0;**

**}**

**b.c**

**#include <stdlib.h>**

**#include <stdio.h>**

**#include <unistd.h>**

**#include <fcntl.h>**

**#include <ctype.h>**

**#include <stdbool.h>**

**int main(int argc, char \*argv[]) { /\*принимаем аргументы командной строки. argc - это количество аргументов командной строки,**

**а argv[] - массив указателей на строки, представляющие собой эти аргументы.\*/**

**int pcb = atoi(argv[1]); /\*извлекаем 2 аргумента и преобразуем их в целочисленные значения с помощью функции atoi()\*/**

**int pab = atoi(argv[2]);**

**size\_t size;**

**while (read(pab, &size, sizeof(size\_t)) > 0) { /\*ждем от А размер\*/**

**/\*как только А завершится, выход из цикла\*/**

**printf("B - из A: %zu\n", size); /\*выводим что получили размер из А\*/**

**read(pcb, &size, sizeof(size\_t)); /\*ждем размер от С\*/**

**printf("B - из C: %zu\n", size); /\*выводим\*/**

**}**

**close(pcb);**

**close(pab);**

**return 0;**

**}**

**c.c**

**#include <stdlib.h>**

**#include <stdio.h>**

**#include <unistd.h>**

**#include <fcntl.h>**

**#include <ctype.h>**

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

**int pac = atoi(argv[1]);**

**int pca = atoi(argv[2]);**

**int pcb = atoi(argv[3]);**

**size\_t size;**

**while (read(pac, &size, sizeof(size\_t)) > 0) { /\*читаем размер строки из пак и выделяем память\*/**

**char \*str = (char\*) malloc(size);**

**if (str == NULL) {**

**printf("Выделение памяти из С\n");**

**exit(-1);**

**}**

**read(pac, str, size); /\*читаем\*/**

**printf("C - из A: %s\n", str); /\*выводим строку\*/**

**write(pcb, &size, sizeof(size\_t)); /\*записываем размер в канал связанный с B\*/**

**int ok = 1; /\*говорим процессу А, что он может читать след строку\*/**

**write(pca, &ok, sizeof(int));**

**free(str);**

**}**

**close(pac);**

**close(pca);**

**close(pcb);**

**return 0;**

**}**

## **Демонстрация работы программы**

imedzhidli@imedzhidli:~/Desktop/OS/KP/build$ ./a

test

C - из A: test

B - из A: 4

B - из C: 4

this is LABA

C - из A: this is LABA

B - из A: 12

B - из C: 12

ʕ ᵔᴥᵔ ʔ

C - из A: ʕ ᵔᴥᵔ ʔ

B - из A: 15

B - из C: 15

٩(｡•́‿•̀｡)۶

C - из A: ٩(｡•́‿•̀｡)۶

B - из A: 25

B - из C: 25

imedzhidli@imedzhidli:~/Desktop/OS/KP/build$

**Выводы:**

Курсовой проект помог мне еще раз на практике применить каналы в качестве механизма взаимодействия между процессами. Также я освежил в памяти процесс создания дочерних процессов и замену образа памяти.