

Московский Авиационный Институт  
(Национальный Исследовательский Университет)  
Факультет информационных технологий и прикладной математики  
Кафедра вычислительной математики и программирования

**Курсовая работа по курсу**  
**«Операционные системы»**  
**III Семестр**  
**Взаимодействие между процессами. Каналы.**

Студент: Боев Савелий Сергеевич  
Группа: М8О-207Б-21  
Преподаватель: Миронов Евгений Сергеевич  
Оценка: \_\_\_\_\_  
Дата: \_\_\_\_\_  
Подпись: \_\_\_\_\_

Москва, 2022

## Содержание

Репозиторий .....	3
Постановка задачи.....	3
Цель работы .....	3
Задание.....	3
Описание идеи задачи.....	4
Общий метод и алгоритм решения.....	5
Исходный код .....	6
a.c.....	6
b.c .....	9
c.c.....	10
Демонстрация работы программы .....	11
Выводы .....	12

# Репозиторий

<https://github.com/IamNoobLEL/Labs-OSi>

## Постановка задачи

### Цель работы

Изучение операционных систем

### Задание

Требуется создать три программы А, В, С. Программа А принимает из стандартного ввода строки, а далее их отправляет программе В. Отправка строк должна производиться построчно. Программа В печатает в стандартный вывод, полученную строку от программы А. После получения программа В отправляет программе А сообщение о том что строка получена. До тех пор пока программа А не получит сообщение о получении строки от программы В, она не может отправлять следующую строку программе В. Программа С пишет в стандартный вывод количество отправленных символов программой А и количество принятых символов программой В. Данную информацию программа С получает от программ А и В соответственно.

## Описание идеи задачи

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. Системные ошибки обработал частично.

## Общий метод и алгоритм решения

Программа содержит функцию, которая считывает строку со стандартного потока вывода.

Также написал функцию, которая возвращает длину строки, она необходима для получения длины входящей строки и последующей передачи полученной длины.

Родитель создаёт два дочерних процесса. В первом потомке закрываем ненужные файловые дескрипторы и выполняем программу В, передав ей необходимые файловые дескрипторы каналов для межпроцессорного взаимодействия.

Второй процесс делает всё тоже самое, что и первый. Родитель передаёт программе С с помощью канала размер входящей строки, саму строку и ожидания получения строки программой С.

После работы всех программ необходимо закрыть все файловые дескрипторы.

# Исходный код

## a.c

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <unistd.h>
4. #include <fcntl.h>
5. #include <ctype.h>
6. #include <stdbool.h>
7.
8. #define MIN_CAP 4
9. #define STDIN 0
10.
11. size_t read_string(char **str_, int fd) {
12.     free(*str_);
13.     size_t str_size = 0;
14.     size_t cap = MIN_CAP;
15.     char *str = (char*) malloc(sizeof(char) * cap);
16.     if (str == NULL) {
17.         perror("Malloc error");
18.         exit(-1);
19.     }
20.     char c;
21.     while (read(fd, &c, sizeof(char)) == 1) {
22.         if (c == '\n') {
23.             break;
24.         }
25.         str[(str_size)++] = c;
26.         if (str_size == cap) {
27.             str = (char*) realloc(str, sizeof(char) * cap * 3 / 2);
28.             cap = cap * 3 / 2;
29.             if (str == NULL) {
30.                 perror("Realloc error");
31.                 exit(-2);
32.             }
33.         }
34.     }
35.     str[str_size] = '\0';
36.
37.     *str_ = str;
38.     return str_size;
39. }
40.
41. size_t str_length(char *str) {
42.     size_t length = 0;
43.     for (int i = 0; str[i] != '\0'; ++i) {
44.         ++length;
45.     }
```

```

46.     return length;
47. }
48.
49. int main() {
50.     int ab[2];
51.     int ac[2];
52.     int ca[2];
53.     int cb[2];
54.
55.
56.     pipe(ab);
57.     pipe(ac);
58.     pipe(ca);
59.     pipe(cb);
60.
61.     int id1 = fork();
62.
63.     if (id1 < 0) {
64.         perror("Fork error");
65.         exit(1);
66.     }
67.     else if (id1 == 0) {
68.         close(ac[1]);
69.         close(ca[0]);
70.         close(cb[0]);
71.         close(ab[0]);
72.         close(ab[1]);
73.
74.         char pac[3];
75.         sprintf(pac, "%d", ac[0]);
76.
77.         char pca[3];
78.         sprintf(pca, "%d", ca[1]);
79.
80.         char pcb[3];
81.         sprintf(pcb, "%d", cb[1]);
82.
83.         execl("./c", "./c", pac, pca, pcb, NULL);
84.     }
85.     else {
86.         int id2 = fork();
87.         if (id2 < 0) {
88.             perror("Fork error");
89.             exit(1);
90.         }
91.         else if (id2 == 0) {
92.             close(ac[0]);
93.             close(ac[1]);
94.             close(ca[0]);
95.             close(ca[1]);
96.             close(cb[1]);

```

```

97.         close(ab[1]);
98.
99.         char pcb[2];
100.            sprintf(pcb, "%d", ca[0]);
101.
102.            char pab[2];
103.            sprintf(pab, "%d", cb[0]);
104.
105.            execl("./b", "./b",  pcb, pab, NULL);
106.        }
107.        else {
108.            close(ac[0]);
109.            close(ca[1]);
110.            close(ab[0]);
111.            close(cb[1]);
112.            close(cb[0]);
113.
114.            char *str = NULL;
115.            while ((read_string(&str, STDIN)) > 0) {
116.                size_t size = str_length(str);
117.                write(ac[1], &size, sizeof(size_t));
118.                write(ac[1], str, size);
119.                write(ab[1], &size, sizeof(size_t));
120.
121.                int ok;
122.                read(ca[0], &ok, sizeof(ok));
123.            }
124.
125.            close(ca[0]);
126.            close(ac[1]);
127.            close(ab[1]);
128.        }
129.    }
130.
131.    return 0;
132. }

```



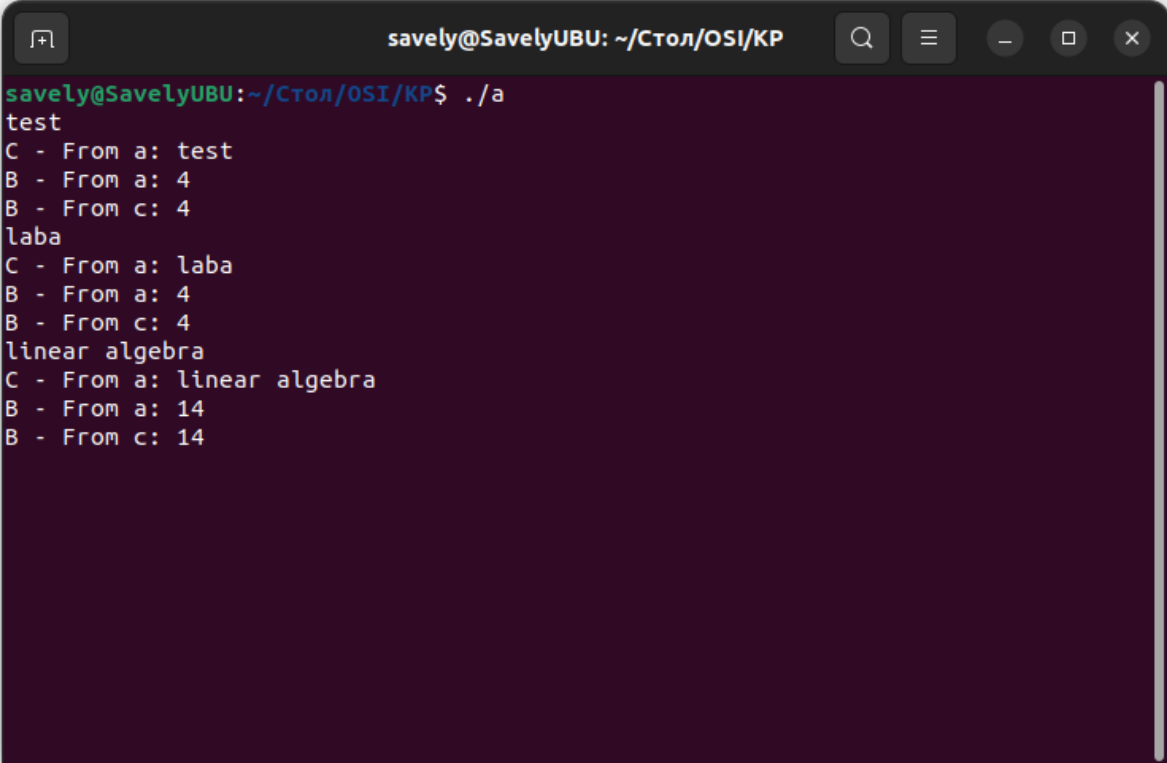
## b.c

```
1. #include <stdlib.h>
2. #include <stdio.h>
3.
4. #include <unistd.h>
5.
6. #include <fcntl.h>
7. #include <ctype.h>
8. #include <stdbool.h>
9.
10. int main(int argc, char *argv[]) {
11.     int pcb = atoi(argv[1]);
12.     int pab = atoi(argv[2]);
13.
14.     size_t size;
15.
16.     while (read(pab, &size, sizeof(size_t)) > 0) { // ждёт от А размер
17.         // как только программа А завершится, выход из while
18.         printf("B - From a: %zu\n", size);
19.         read(pcb, &size, sizeof(size_t));
20.         printf("B - From c: %zu\n", size);
21.     }
22.
23.     close(pcb);
24.     close(pab);
25.
26.     return 0;
27. }
```

## c.c

```
1. #include <stdlib.h>
2. #include <stdio.h>
3. #include <unistd.h>
4. #include <fcntl.h>
5. #include <ctype.h>
6.
7.
8. int main(int argc, char *argv[]) {
9.     // argv[0] = ".\c";
10.    // argv[1] = 3
11.    // argv[2] = 4
12.    // argv[3] = 5
13.
14.    // atoi: строку переводит в int
15.    int pac = atoi(argv[1]);
16.    int pca = atoi(argv[2]);
17.    int pcb = atoi(argv[3]);
18.
19.    size_t size;
20.
21.    while (read(pac, &size, sizeof(size_t)) > 0) { // ждёт от программы А размер
22.        char *str = (char*) malloc(size); //
23.        if (str == NULL) {
24.            printf("MALLOC from C\n");
25.            exit(-1);
26.        }
27.        read(pac, str, size); // ждёт от программы А строку
28.        printf("C - From a: %s\n", str);
29.        write(pcb, &size, sizeof(size_t));
30.        int ok = 1;
31.        write(pca, &ok, sizeof(int));
32.        free(str);
33.    }
34.
35.    close(pac);
36.    close(pca);
37.    close(pcb);
38.
39.
40.    return 0;
41. }
```

## Демонстрация работы программы



```
savely@SavelyUBU: ~/Стол/OSI/KP
savely@SavelyUBU:~/Стол/OSI/KP$ ./a
test
C - From a: test
B - From a: 4
B - From c: 4
laba
C - From a: laba
B - From a: 4
B - From c: 4
linear algebra
C - From a: linear algebra
B - From a: 14
B - From c: 14
```

## **Выводы**

Данный курсовой проект направлен на изучение и применение на практике одного из механизмов межпроцессорного взаимодействия, такого как, каналы, отображение файла в оперативную память или очередь сообщений.

В целом, курсовой проект не вызвал у меня каких-либо трудностей, так как мною были приобретены теоретические сведения во время выполнения второй лабораторной работы по курсу «Операционные системы».

Однако, курсовой проект помог мне на практике применить каналы в качестве механизма взаимодействия между процессами, а также освежил в памяти создание процессов и замену образа памяти.