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: Селивёрстов Д.С.

Группа: М8О-201Б-21

Преподаватель: Миронов Е. С.

Вариант 39

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2022

**Содержание**

[Постановка задачи 3](#_Toc123906326)

[Цель работы 3](#_Toc123906327)

[Задание 3](#_Toc123906328)

[Описание идеи задачи 4](#_Toc123906329)

[Общий метод и алгоритм решения 5](#_Toc123906330)

[Исходный код 6](#_Toc123906331)

[a.c 6](#_Toc123906332)

[b.c 9](#_Toc123906333)

[c.c 10](#_Toc123906334)

[Демонстрация работы программы 11](#_Toc123906335)

[Выводы 12](#_Toc123906336)

# **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 полученяя длину 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 делаёт 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>
8. #define MIN\_CAP 4
9. #define STDIN 0
11. size\_t read\_string(char \*\*str\_, int fd) {
12. [free](http://www.opengroup.org/onlinepubs/009695399/functions/free.html)(\*str\_);
13. size\_t str\_size = 0;
14. size\_t cap = MIN\_CAP;
15. char \*str = (char\*) [malloc](http://www.opengroup.org/onlinepubs/009695399/functions/malloc.html)(sizeof(char) \* cap);
16. if (str == NULL) {
17. [perror](http://www.opengroup.org/onlinepubs/009695399/functions/perror.html)("Malloc error");
18. [exit](http://www.opengroup.org/onlinepubs/009695399/functions/exit.html)(-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](http://www.opengroup.org/onlinepubs/009695399/functions/realloc.html)(str, sizeof(char) \* cap \* 3 / 2);
28. cap = cap \* 3 / 2;
29. if (str == NULL) {
30. [perror](http://www.opengroup.org/onlinepubs/009695399/functions/perror.html)("Realloc error");
31. [exit](http://www.opengroup.org/onlinepubs/009695399/functions/exit.html)(-2);
32. }
33. }
34. }
35. str[str\_size] = '**\0**';
37. \*str\_ = str;
38. return str\_size;
39. }
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. }
49. int main() {
50. int ab[2];
51. int ac[2];
52. int ca[2];
53. int cb[2];

56. pipe(ab);
57. pipe(ac);
58. pipe(ca);
59. pipe(cb);
61. int id1 = fork();
63. if (id1 < 0) {
64. [perror](http://www.opengroup.org/onlinepubs/009695399/functions/perror.html)("Fork error");
65. [exit](http://www.opengroup.org/onlinepubs/009695399/functions/exit.html)(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]);
74. char pac[3];
75. [sprintf](http://www.opengroup.org/onlinepubs/009695399/functions/sprintf.html)(pac, "%d", ac[0]);
77. char pca[3];
78. [sprintf](http://www.opengroup.org/onlinepubs/009695399/functions/sprintf.html)(pca, "%d", ca[1]);
80. char pcb[3];
81. [sprintf](http://www.opengroup.org/onlinepubs/009695399/functions/sprintf.html)(pcb, "%d", cb[1]);
83. execl("./c", "./c", pac, pca, pcb, NULL);
84. }
85. else {
86. int id2 = fork();
87. if (id2 < 0) {
88. [perror](http://www.opengroup.org/onlinepubs/009695399/functions/perror.html)("Fork error");
89. [exit](http://www.opengroup.org/onlinepubs/009695399/functions/exit.html)(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]);
99. char pcb[2];
100. [sprintf](http://www.opengroup.org/onlinepubs/009695399/functions/sprintf.html)(pcb, "%d", ca[0]);
102. char pab[2];
103. [sprintf](http://www.opengroup.org/onlinepubs/009695399/functions/sprintf.html)(pab, "%d", cb[0]);
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]);
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));
121. int ok;
122. read(ca[0], &ok, sizeof(ok));
123. }
125. close(ca[0]);
126. close(ac[1]);
127. close(ab[1]);
128. }
129. }
131. return 0;
132. }

## **b.c**

1. #include <stdlib.h>
2. #include <stdio.h>
4. #include <unistd.h>
6. #include <fcntl.h>
7. #include <ctype.h>
8. #include <stdbool.h>
10. int main(int argc, char \*argv[]) {
11. int pcb = [atoi](http://www.opengroup.org/onlinepubs/009695399/functions/atoi.html)(argv[1]);
12. int pab = [atoi](http://www.opengroup.org/onlinepubs/009695399/functions/atoi.html)(argv[2]);
14. size\_t size;
16. while (read(pab, &size, sizeof(size\_t)) > 0) { *// ждёт от А размер*
17. *// как только программа А завершится, выход из while*
18. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("B - From a: %zu**\n**", size);
19. read(pcb, &size, sizeof(size\_t));
20. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("B - From c: %zu**\n**", size);
21. }
23. close(pcb);
24. close(pab);
26. return 0;
27. }

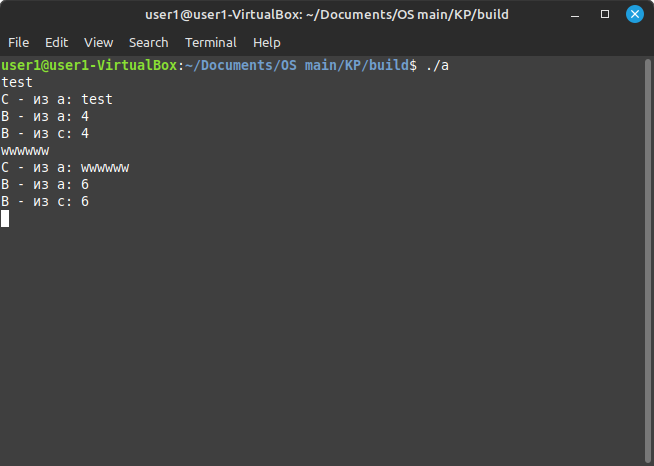
## **c.c**

1. #include <stdlib.h>
2. #include <stdio.h>
3. #include <unistd.h>
4. #include <fcntl.h>
5. #include <ctype.h>

8. int main(int argc, char \*argv[]) {
9. *// argv[0] = ".\c";*
10. *// argv[1] = 3*
11. *// argv[2] = 4*
12. *// argv[3] = 5*
14. *// atoi: строку переводит в int*
15. int pac = [atoi](http://www.opengroup.org/onlinepubs/009695399/functions/atoi.html)(argv[1]);
16. int pca = [atoi](http://www.opengroup.org/onlinepubs/009695399/functions/atoi.html)(argv[2]);
17. int pcb = [atoi](http://www.opengroup.org/onlinepubs/009695399/functions/atoi.html)(argv[3]);
19. size\_t size;
21. while (read(pac, &size, sizeof(size\_t)) > 0) { *// ждёт от программы А размер*
22. char \*str = (char\*) [malloc](http://www.opengroup.org/onlinepubs/009695399/functions/malloc.html)(size); *//*
23. if (str == NULL) {
24. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("MALLOC from C**\n**");
25. [exit](http://www.opengroup.org/onlinepubs/009695399/functions/exit.html)(-1);
26. }
27. read(pac, str, size); *// ждёт от программы А строку*
28. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("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](http://www.opengroup.org/onlinepubs/009695399/functions/free.html)(str);
33. }
35. close(pac);
36. close(pca);
37. close(pcb);

40. return 0;
41. }

## **Демонстрация работы программы**



# **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.