Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Курсовая работа**

**по курсу "Операционные системы"**

**3 семестр**

**Вариант на удовлетворительно**

Студент: Гречников А.В.

Группа: М8О–206Б–20

Преподаватель: Соколов Андрей Алексеевич

Дата: 12.01.2022

Оценка: 3

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2021

**Постановка задачи**

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 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. 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 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 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.

**Основные файлы программы**

a.c:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#include <ctype.h>

#define MIN\_CAP 1

#define STDIN 0

size\_t read\_string(char \*\*str\_, int fd) {

free(\*str\_);

size\_t str\_size = 0;

size\_t cap = MIN\_CAP;

char \*str = (char\*) malloc(sizeof(char) \* cap);

if (str == NULL) {

perror("Malloc error");

exit(-1);

}

char c;

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

if (c == '\n') {

break;

}

str[(str\_size)++] = c;

if (str\_size == cap) {

str = (char\*) realloc(str, sizeof(char) \* cap \* 3 / 2);

cap = cap \* 3 / 2;

if (str == NULL) {

perror("Realloc error");

exit(-2);

}

}

}

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

\*str\_ = str;

return str\_size;

}

size\_t str\_length(char \*str) {

size\_t length = 0;

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

++length;

}

return length;

}

void pipe\_wrapper(int pipefd[2]) {

if (pipe(pipefd) == -1) {

perror("Cannot create pipe");

exit(1);

}

}

int main() {

int ab[2];

int ac[2];

int ca[2];

int cb[2];

pipe\_wrapper(ab);

pipe\_wrapper(ac);

pipe\_wrapper(ca);

pipe\_wrapper(cb);

int id1 = fork();

if (id1 < 0) {

perror("Fork error");

exit(1);

}

else if (id1 == 0) {

close(ac[1]);

close(ca[0]);

close(cb[0]);

close(ab[0]);

close(ab[1]);

char pac[3];

sprintf(pac, "%d", ac[0]);

char pca[3];

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

char pcb[3];

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

execl("./c", "./c", pac, pca, pcb, NULL);

}

else {

int id2 = fork();

if (id2 < 0) {

perror("Fork error");

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]);

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) {

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;

}

c.c:

#include <stdlib.h>

#include <stdio.h>

#include <unistd.h>

#include <fcntl.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) {

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

if (str == NULL) {

printf("MALLOC from C\n");

}

read(pac, str, size);

printf("[C] - From A: %s\n", str);

write(pcb, &size, sizeof(size\_t));

int ok = 1;

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

free(str);

}

close(pac);

close(pca);

close(pcb);

return 0;

}

b.c:

#include <stdlib.h>

#include <stdio.h>

#include <unistd.h>

#include <fcntl.h>

#include <ctype.h>

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

int pcb = atoi(argv[1]);

int pab = atoi(argv[2]);

size\_t size;

while (read(pab, &size, sizeof(size\_t)) > 0) {

printf("[B] - Get A: %zu\n", size);

read(pcb, &size, sizeof(size\_t));

printf("[B] - Get C: %zu\n", size);

}

close(pcb);

close(pab);

return 0;

}

Makefile:

CC=gcc

CFLAGS=-pedantic -Wall

all: a b c

a: a.o

gcc a.c -o a

b: b.o

gcc b.c -o b

c: c.o

gcc c.c -o c

clean:

rm a b c \*.o

**Пример работы**

artem@ideapad:~$ ./a

qwerty

[C] - From A: qwerty

[B] - Get A: 6

[B] - Get C: 6

wasd

[C] - From A: wasd

[B] - Get A: 4

[B] - Get C: 4

aab

[C] - From A: aab

[B] - Get A: 3

[B] - Get C: 3

**Вывод**

Для написания данного курсового проекта я использовал знания, полученные за этот курс. Способом межпроцессорного взаимодействия я выбрал однонаправленные каналы, о которых узнал в лабораторной работе 2. Благодаря курсовому проекту я смог применить на практике каналы, а также смог освежить в памяти создание дочерних процессов и работу с ними.