Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №1 по курсу**

**«Операционные системы»**

**Тема работы**

**“Взаимодействие между процессами”**

Студент: Ковриженков Дмитрий Олегович

Группа: М8О-203Б-23

Вариант:13

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

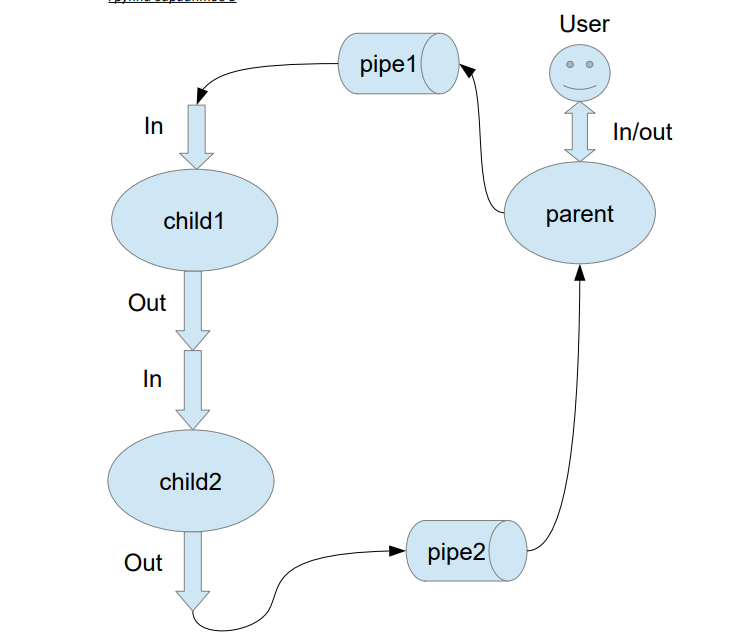
Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2024

**Постановка задачи**

**Лабораторная работа №1**

**Задача:** Родительский процесс создает два дочерних процесса. Перенаправление стандартных потоков ввода-вывода показано на картинке выше. Child1 и Child2 можно «соединить» между собой дополнительным каналом. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1. Процесс child1 и child2 производят работу над строками. Child2 пересылает результат своей работы родительскому процессу. Родительский процесс полученный результат выводит в стандартный поток вывода.



В 13 варианте Child1 переводит строки в нижний регистр. Child2 превращает все пробельные символы в символ «\_».

**Общие сведения**

В данной программе реализуется межпроцессное взаимодействие с использованием каналов (pipe) и создания дочерних процессов (fork). Родительский процесс принимает строку от пользователя и передает её в pipe1, связанный с первым дочерним процессом (child1). Дочерний процесс child1 считывает строку из pipe1, выполняет преобразование символов в нижний регистр с помощью функции To\_lower\_case и передает результат во второй канал pipe3. Затем второй дочерний процесс child2 получает строку из pipe3, заменяет пробелы на символы подчеркивания (Replace\_spaces) и отправляет обработанную строку в pipe2. После завершения работы обоих дочерних процессов родительский процесс получает итоговую строку из pipe2 и выводит её в стандартный вывод. Программа использует fork() для создания процессов, pipe() для организации обмена данными и wait() для синхронизации.

**Вывод**

В процессе разработки данной программы были изучены и применены различные методы межпроцессного взаимодействия. Функция fork() использовалась для создания дочерних процессов, pipe() – для организации потоков передачи данных, а dup2() (в данном случае не используется, но мог бы применяться) позволил бы перенаправлять стандартные потоки ввода-вывода. Также была реализована передача данных между процессами через write() и read(), а синхронизация процессов осуществлялась с помощью wait().

Приложение

src/child1.c

#include "child.h"

#include "utils.h"

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

void ChildRoutine(int pipe\_cin[2], int pipe\_out[2], void (\*func)(char \*str)) {

char buffer[BUFFER\_SIZE];

close(pipe\_cin[1]);

close(pipe\_out[0]);

if (read(pipe\_cin[0], buffer, BUFFER\_SIZE) == -1) {

perror("Ошибка при чтении из pipe");

exit(1);

}

func(buffer);

if (write(pipe\_out[1], buffer, strlen(buffer) + 1) == -1) {

perror("Ошибка при записи в pipe");

exit(1);

}

close(pipe\_cin[0]);

close(pipe\_out[1]);

}

src/parent.c

#include "parent.h"

#include "child.h"

#include "utils.h"

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <sys/wait.h>

char\* ParentRoutine(const char\* inputString) {

int pipe1[2];

int pipe2[2];

int pipe3[2];

static char buffer[BUFFER\_SIZE];

Create\_pipe(pipe1);

Create\_pipe(pipe2);

Create\_pipe(pipe3);

pid\_t child1 = fork();

if (child1 == -1) {

perror("Ошибка при вызове fork для Child1");

exit(1);

}

if (child1 == 0) {

ChildRoutine(pipe1, pipe3, To\_lower\_case);

exit(0);

}

pid\_t child2 = fork();

if (child2 == -1) {

perror("Ошибка при вызове fork для Child2");

exit(1);

}

if (child2 == 0) {

ChildRoutine(pipe3, pipe2, Replace\_spaces);

exit(0);

}

close(pipe1[0]);

close(pipe3[0]);

close(pipe3[1]);

close(pipe2[1]);

strncpy(buffer, inputString, BUFFER\_SIZE - 1);

buffer[BUFFER\_SIZE - 1] = '\0';

if (write(pipe1[1], buffer, strlen(buffer) + 1) == -1) {

perror("Ошибка при записи в pipe1");

exit(1);

}

close(pipe1[1]);

wait(NULL);

wait(NULL);

if (read(pipe2[0], buffer, BUFFER\_SIZE) == -1) {

perror("Ошибка при чтении из pipe2");

exit(1);

}

close(pipe2[0]);

return buffer;

}

src/utils.c

#include "utils.h"

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <string.h>

void Create\_pipe(int pipeFd[2]) {

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

perror("Ошибка при создании pipe");

exit(1);

}

}

void To\_lower\_case(char \*str) {

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

str[i] = tolower((unsigned char)str[i]);

}

}

void Replace\_spaces(char \*str) {

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

if (str[i] == ' ') {

str[i] = '\_';

}

}

}

include/ child.h

#ifndef CHILD\_H

#define CHILD\_H

void ChildRoutine(int pipe\_cin[2], int pipe\_out[2], void (\*func)(char \*str));

#endif

include / parent.h

#ifndef PARENT\_H

#define PARENT\_H

#define BUFFER\_SIZE 256

#ifdef \_\_cplusplus

extern "C" {

#endif

char\* ParentRoutine(const char\* inputString);

#ifdef \_\_cplusplus

}

#endif

#endif

include / utils.h

#ifndef UTILS\_H

#define UTILS\_H

#define BUFFER\_SIZE 256

void Create\_pipe(int pipeFd[2]);

void To\_lower\_case(char \*str);

void Replace\_spaces(char \*str);

#endif

main.c

#include "parent.h"

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main(void) {

char buffer[BUFFER\_SIZE];

printf("Введите строку: ");

if (fgets(buffer, BUFFER\_SIZE, stdin) == NULL) {

perror("Ошибка при вводе данных");

exit(1);

}

buffer[strcspn(buffer, "\n")] = '\0';

const char\* result = ParentRoutine(buffer);

printf("Результат : %s\n", result);

return 0;

}

**CMakeLists.txt**

cmake\_minimum\_required(VERSION 3.10)

project(lab1 C)

set(CMAKE\_C\_STANDARD 11)

add\_executable(lab1

main.c

src/parent.c

src/child.c

src/utils.c)

target\_include\_directories(lab1 PRIVATE include)

Пример вывода:  
