Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №2 по курсу**

**«Операционные системы»**

**Процессы операционных систем**

Студент: Армишев Кирилл Константинович

Группа: М8О–208Б–21

Вариант: 12

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2022.

**Постановка задачи**

## Цель работы

Целью является приобретение практических навыков в:

* Управление процессами в ОС
* Обеспечение обмена данных между процессами посредством каналов

## Задание

Составить и отладить программу на языке Си, осуществляющую работу с процессами и

взаимодействие между ними в одной из двух операционных систем. В результате работы

программа (основной процесс) должен создать для решение задачи один или несколько

дочерних процессов. Взаимодействие между процессами осуществляется через системные

сигналы/события и/или каналы (pipe).

Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

**Общие сведения о программе**

Программа компилируется из файла main.c. Также используется заголовочные файлы: <stdio.h>, <ctype.h>, <string.h, <unistd.h>. В программе используются следующие системные вызовы:

1. **pid\_t fork()** - создание дочернего процесса, возвращает -1 при ошибке создания дочернего процесса, 0 если процесс является дочерним, и pid если процесс является родительским.
2. **int execlp(const char \*file, const char \*arg, ...)** – заменяет текущий образ процесса новым образом процесса, c аргументами arg.
3. **int pipe(int pipefd[2])** - создание неименованного канала для передачи данных между процессами
4. **int dup2(int oldfd, int newfd)** - переназначение файлового дескриптора
5. **int close(int fd)** - закрыть файл

**Общий метод и алгоритм решения**.

Для реализации поставленной задачи необходимо:

1. Изучить принципы работы fork, pipe, execlp, dup2.
2. Написать программу child1 для перевода текста в верхний регистр
3. Написать программу child2 для замены двух и более пробельных символов на один пробел
4. Написать программу создающую два дочерних процесса заменяемые child1 и child2 и соединить каналами pipe1 и pipe2

**Основные файлы программы**

**main.c:**

#include <stdio.h>  
#include <ctype.h>  
#include <string.h>  
#include <unistd.h>  
  
int main(){  
 int fd1[2];  
 if (pipe(fd1)==-1){  
 perror("error pipe1");  
 }  
 int fd2[2];  
 if (pipe(fd2)==-1){  
 perror("error pipe2");  
 }  
 int fd3[2];  
 if (pipe(fd3)==-1){  
 perror("error pipe3");  
 }  
 int id = fork();  
 if (id == -1)  
 {  
 perror("fork error");  
 return 0;  
 }else if(id == 0) {  
 close(fd1[1]);  
 close(fd2[0]);  
 close(fd3[0]);  
 close(fd3[1]);  
 dup2(fd1[0], STDIN\_FILENO);  
 dup2(fd2[1], STDOUT\_FILENO);  
 execlp("./child1", "child1", NULL);  
 close(fd1[0]);  
 close(fd2[1]);  
 }  
 int id2;  
 if(id > 0) {  
 id2 = fork();  
 if (id2 == -1)  
 {  
 perror("fork2 error");  
 return 0;  
 }else if(id2 == 0) {  
 close(fd3[0]);  
 close(fd2[1]);  
 close(fd1[0]);  
 close(fd1[1]);  
 dup2(fd2[0], STDIN\_FILENO);  
 dup2(fd3[1], STDOUT\_FILENO);  
 execlp("./child2", "child2", NULL);  
 close(fd2[0]);  
 close(fd3[1]);  
 }  
 }  
 if(id != 0 && id2 != 0) {  
 close(fd1[0]);  
 close(fd2[0]);  
 close(fd2[1]);  
 close(fd3[1]);  
 char c[100];  
 while(fgets(c,sizeof(c),stdin)) {  
 int n = strlen(c);  
 write(fd1[1], &c, n);  
 int n1= read(fd3[0], &c, n);  
 write(1,&c,n1);  
 }  
 close(fd3[0]);  
 close(fd1[1]);  
 }  
  
 return 0;  
}

**child1.c**

#include<unistd.h>  
#include <ctype.h>  
int main() {  
 char c[100];  
 int n;  
 while(n=read(0, &c, 100)) {  
 for (int i = 0; i <= n; ++i) {  
 if (c[i] != '\n' && c[i] != '\0') {  
 c[i] = toupper(c[i]);  
 } else {  
 break;  
 }  
 }  
 write(1, &c, n);  
 }  
 return 0;  
}

**child2.c**

#include<unistd.h>  
#include <ctype.h>  
void \*spaces(char\* str) {  
 int i, x;  
 for(i=x=0; str[i]; ++i)  
 if(!isspace(str[i]) || (i > 0 && !isspace(str[i-1])))  
 str[x++] = str[i];  
 str[x] = '\0';  
 return str;  
}  
  
int main() {  
 char c[100];  
 while (read(0, &c, 100)) {  
 spaces(c);  
 size\_t k = sizeof(c)/sizeof(c[0]);  
 write(1, c, k);  
 }  
}

**Пример работы**

**1.txt:**

kgktlgfkfk ddldld dkdkdk  
fed dedede dededede de

**Output:**

-------------------------test 1-------------------------

KGKTLGFKFK DDLDLD DKDKDK

FED DEDEDE DEDEDEDE DE

**Вывод**

В результате данной лабораторной работы были изучены основные методы работы с процессами в ОС linux. В данной работе я научился создавать процессы, работать с родительскими и дочерними процессами, передавать между ними данные с помощью каналов. Полученные знания пригодятся не только при выполнении дальнейших лабораторных работ, но и в дальнейшей работе.