Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №2 по курсу**

**«Операционные системы»**

Студент: Гордовой Д.С.

Группа: М8О–201Б–19

Вариант: 13

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2020.

**Постановка задачи**

**Цель работы**

Приобретение практических навыков в:

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

**Задание**

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.

**Вариант задания**

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

**Общие сведения о программе**

Программа компилируется из файла lab2.c. . В программе используются следующие системные вызовы:

1. **fork** - создает новый [процесс](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) (потомок), который является практически полной копией процесса-родителя, выполняющего этот вызов.
2. **pipe** - создаёт однонаправленный канал данных, который можно использовать для взаимодействия между процессами.
3. **read** - Считывает данные из файла
4. **write -**  записывает в файл

**Код программы**

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

*int* main()

{

*int* fd\_0[2]; *// три массива*

*int* fd\_1[2];

*int* fd\_2[2];

pipe(fd\_0); *// три трубы создаем без проверки на ошибку*

pipe(fd\_1);

pipe(fd\_2); *// треться труба между вторым чайлдом и родителем*

*int* pid\_0;

*int* pid\_1;

*static char* output[50];

*if*((pid\_0 = fork()) > 0) *// после разъединения если ты родитель*

{

*if*((pid\_1 = fork()) > 0) *// если все еще родитель ты. Родитель может быть один только*

*// то мы делимся еще раз в ребенке*

{

close(fd\_0[0]);

close(fd\_2[1]);

*static char* input[50];

printf("Введите сообщение от родителя: ");

*for*(*int* i=0; (input[i]=getchar())!=**EOF** && input[i]!='\n' && i<49; i++);

write(fd\_0[1], input, 50); *// для первого ребенка пишем в пайп*

close(fd\_0[1]); *// закрываем на запись*

read(fd\_2[0], input, 50); *// читает от второго ребенка*

*for*(*int* i=0; i<50; i++){

output[i] = input[i];

}

printf("В сообщении от ребенка: %s\n", output);

close(fd\_2[0]);

}

*else if*(pid\_1 == 0) *// это для второго ребенка так как это после уже одного форка происходит*

{

close(fd\_1[1]);

close(fd\_2[0]);

*static char* input[50];

*//close(fd[1]);*

read(fd\_1[0], input, 50);

*for*(*int* i = 0; i<49; ++i)

{

*if*(input[i] == ' '){

input[i] = '\_';

}

}

write(fd\_2[1], input, 50);

close(fd\_2[1]);

close(fd\_1[0]);

}

*else*

{

perror("fork error\n");

exit(-1);

}

}

*else if* (pid\_0 == 0) *// это для первого ребенка. Так как мы поделили уже один раз форком*

{

close(fd\_0[1]);

close(fd\_1[0]);

*static char* input[50];

*//close(fd[1]);*

read(fd\_0[0], input, 50);

*for*(*int* i = 0; i<49; ++i)

{

input[i] = tolower(input[i]); *// tolower это функция из библиотеки <ctype.h> которая переводит в нижний регистр буквы*

}

write(fd\_1[1], input, 50);

close(fd\_0[0]);

close(fd\_1[1]);

}

*else*

{

perror("fork error\n");

exit(-1);

}

*return* 0;

}

**Пример работы**

pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~$ gcc lab2\_nix.c

pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~$ cat test.txt

heLlo woRld

gooDbye tyna NonaMe

reaD my Prooggma

Typak NonE gooD

EEeee enD thnks!

pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~$ ./a.out < test.txt

HELLO WORLD

GOODBYE TYNA NONAME

READ MY PROOGGMA

TYPAK NONE GOOD

EEEEE END THNKS!

pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~$ ./a.out

HeLllo my friend yeee!

Go playIIIng wIth me

HELLLO MY FRIEND YEEE!

GO PLAYIIING WITH ME

**Вывод**

В данной лабораторной удалось познакомиться с таким системным вызовом как fork() для создания новых процессов и с pipe который служит для их связи этих процессов. Эти команды могут усложнить код программы но зато уменьшить время ее работы, что является одним из главных задач программиста при написании программы