Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №2 по курсу**

**«Операционные системы»**

**МЕЖПРОЦЕССНОЕ ВЗАИМОДЕЙСТВИЕ**

Студент: Уваров Андрей Павлович

Группа: М8О–206Б–20

Вариант: 19

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2020.

**Постановка задачи**

## Цель работы

Целью работы 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.

Вариант 19:

Родительский процесс создает дочерний процесс. В первых двух строчках пользователь вводит названия двух файлов, которые будут созданы для записи в них данных из дочерних процессов.

Родительский процесс передает строки пользователя в процентном соотношении (80/20) через fd или fd2 первому и второму дочернему процессу соответственно. Результаты своей работы дочерний процесс пишет в созданный им файл. Пользователь вводит строки вида: «строка\n строка\n строка\n Ctrl+D». Далее эти строки передаются от родительского процесса в один из дочерних. Дочерний процесс вызывает функцию, которая создает новую строку и в нее записывает все согласные буквы исходной строки, потом выводит в файл.

**Общие сведения о программе**

Программа компилируется из файла var\_19\_main.c. Также используется заголовочные файлы:

<unistd.h>,<stdlib.h>,<sys/types.h>,<sys/stat.h>,<fcntl.h>,<time.h>,"lab\_2.h".

В программе используются следующие системные вызовы:

1. **pipe** ­­–– принимает массив из двух целых чисел, в случае успеха массив будет содержать два файловых дескриптора, которые будут использоваться для конвейера, первое число в массиве предназначено для чтения, второе для записи, а так же вернется 0. В случае неуспеха вернется -1.
2. **fork ––** создает новый процесс, который является копией родительского процесса, за исключением разных process ID и parent process ID. В случае успеха fork() возвращает 0 для ребенка, число больше 0 для родителя – child ID, в случае ошибки возвращает -1.
3. **close ––** принимает файловый дескриптор в качестве аргумента, удаляет файловый дескриптор из таблицы дескрипторов, в случае успеха вернет 0, в случае неуспеха вернет -1.
4. **open ––** создает или открывает файл, если он был создан. В качестве аргументов принимает путь до файла, режим доступа (запись, чтение и т.п.), модификатор доступа ( при создании можно указать права для файла ). Возвращает в случае успеха файловый дескриптор – положительное число, иначе возвращает -1.
5. **read ––** предназначена для чтения какого-то числа байт из файла, принимает в качестве аргументов файловый дескриптор, буфер, в который будут записаны данные и число байт. В случае успеха вернет число прочитанных байт, иначе -1.
6. **write ––** предназначена для записи какого-то числа байт в файл, принимает в качестве аргументов файловый дескриптор, буфер, из которого будут считаны данные для записи и число байт. В случае успеха вернет число записанных байт, иначе -1.

**Общий метод и алгоритм решения**.

Для реализации поставленной задачи необходимо:

1. Изучить принципы работы pipe, fork.
2. Реализовать функции для процесса-родителя и процесса-ребенка.
3. Реализовать сообщение между процессами при помощи каналов.
4. Реализовать обработку системных ошибок согласно заданию.

**Основные файлы программы**

**var\_19\_main.c:**

|  |
| --- |
| /\* |
|  | Вариант 19: |
|  | Родитель считывает строки и кидает в соотношении 80% на 20% в первого или второго ребенка соответственно. |
|  | Дети выводят строку в два файла, введеных из родителя, убираю из каждой строки гласные. |
|  | \*/ |
|  |  |
|  | #include <unistd.h> |
|  | #include <stdlib.h> |
|  | #include <sys/types.h> |
|  | #include <sys/stat.h> |
|  | #include <fcntl.h> |
|  | #include <time.h> |
|  | #include "lab\_2.h" |
|  |  |
|  | const int SIZE\_OF\_CHARS = 200; |
|  |  |
|  | int main(){ |
|  | srand(time(0)); |
|  |  |
|  | char path1[200] = "",path2[200] = ""; |
|  |  |
|  | char t[200] = "Введи в следующий двух строках - пути к файлам, куда я буду отправлять выходные данные ^\_^\n"; |
|  | write(0,t,sizeof(char) \* 200); |
|  |  |
|  | read(0,path1,sizeof(char)\*200); |
|  | read(0,path2,sizeof(char)\*200); |
|  |  |
|  | int path1Len = strlen(path1); |
|  | int path2Len = strlen(path2); |
|  |  |
|  | path1[path1Len-1] = '\0'; |
|  | path2[path2Len-1] = '\0'; |
|  |  |
|  | int check = open(path1,O\_CREAT,S\_IRWXU); |
|  | int check2 = open(path2,O\_CREAT,S\_IRWXU); |
|  |  |
|  | if(check == -1 || check2 == -1){ |
|  | char k[200] = "Something wrong!\n"; |
|  | write(0,k,sizeof(char) \* 200); |
|  | return 1; |
|  | } |
|  |  |
|  | int ft1 = open(path1,O\_WRONLY); |
|  | int ft2 = open(path2,O\_WRONLY); |
|  |  |
|  | if(ft1 == -1 || ft2 == -1){ |
|  | char k[200] = "Something wrong!\n"; |
|  | write(0,k,sizeof(char) \* 200); |
|  | return 1; |
|  | } |
|  |  |
|  | int fd[2]; |
|  | int fd2[2]; |
|  | if(pipe(fd) < 0 || pipe(fd2) < 0){ |
|  | char a[50] = "Проблемы с пайпами\n"; |
|  | write(0,a,50\*sizeof(char)); |
|  | return 1; |
|  | } |
|  |  |
|  | int id = fork(); |
|  | int id2 = fork(); |
|  |  |
|  | char in[SIZE\_OF\_CHARS]; |
|  | if(id != 0 && id2 != 0){ |
|  | char a[50] = "Введите свои строки\n"; |
|  | write(0,a,50\*sizeof(char)); |
|  | while((scanf("%s",in)) == 1){ |
|  | fflush(stdout); |
|  | if(rand() % 100 < 80){ |
|  | //printf("First - "); |
|  | char a[50] = "First file\n"; |
|  | write(0,a,50\*sizeof(char)); |
|  | write(fd[1],in,SIZE\_OF\_CHARS \* sizeof(char)); |
|  | } |
|  | else{ |
|  | char a[50] = "Second file\n"; |
|  | write(0,a,50\*sizeof(char)); |
|  | write(fd2[1],in,SIZE\_OF\_CHARS \* sizeof(char)); |
|  | } |
|  | } |
|  |  |
|  | char g = '\*'; |
|  | write(fd[1],&g,sizeof(char)); |
|  | write(fd2[1],&g,sizeof(char)); |
|  | } |
|  | else if(id2 == 0 && id != 0){ |
|  | while(1){ |
|  | fflush(stdout); |
|  | char str[SIZE\_OF\_CHARS]; |
|  | read(fd2[0],str,SIZE\_OF\_CHARS \* sizeof(char)); |
|  | if(str[0] == '\*'){ |
|  | break; |
|  | } |
|  | write(ft2,Task(str),sizeof(char) \* 200); |
|  | } |
|  | } |
|  | else if(id == 0 && id2 != 0){ |
|  | while(1){ |
|  | fflush(stdout); |
|  | char str[SIZE\_OF\_CHARS]; |
|  | read(fd[0],str,SIZE\_OF\_CHARS \* sizeof(char)); |
|  | if(str[0] == '\*'){ |
|  | break; |
|  | } |
|  | write(ft1,Task(str),sizeof(char) \* 200); |
|  | } |
|  | } |
|  |  |
|  | close(fd[0]); |
|  | close(fd[1]); |
|  | close(fd2[0]); |
|  | close(fd2[1]); |
|  | close(ft1); |
|  | close(ft2); |
|  | return 0; |
|  | } |

**lab\_2\_string.c:**

|  |
| --- |
| #include"lab\_2.h" |
|  | #include "malloc.h" |
|  |  |
|  | char \* Task(char \* l){ |
|  | char \* res = malloc(sizeof(char) \* 200); |
|  | int curId = 0; |
|  | for (int i = 0; i < strlen(l); i++) |
|  | { |
|  | char cur = l[i]; |
|  | if(!(cur == 'a' || cur == 'A' || cur == 'u' || cur == 'U' || cur == 'y' || cur == 'Y' || cur == 'o' || cur == 'O' || cur == 'i' || cur == 'I' || cur == 'e' || cur == 'E')){ |
|  | res[curId] = cur; |
|  | curId++; |
|  | } |
|  | } |
|  |  |
|  | int resL = strlen(res); |
|  | res[resL] = '\n'; |
|  | return res; |
|  | } |

**Пример работы**

andrey@andrey-VivoBook:~/cppProjects/OS\_1$ make

gcc var\_19\_main.c lab\_2\_string.c lab\_2.h

andrey@andrey-VivoBook:~/cppProjects/OS\_1$ ./a.out

Введи в следующий двух строках - пути к файлам, куда я буду отправлять выходные данные ^\_^

a

b

Введите свои строки

first

second

third

fourth

fifth

sixth

seventh

vosem

devyat

andrey@andrey-VivoBook:~/cppProjects/OS\_1$ cat a

frst

scnd

thrd

frth

ffth

sxth

svnth

vsm

andrey@andrey-VivoBook:~/cppProjects/OS\_1$ cat b

dvt

**Вывод**

Делая данную лабораторную работу, я познакомился с такими вещами, как процессы и каналы. Связанные процессы могут помочь повысить эффективность программы, решающей задачу, в которой есть множество независимых действий. Я понял как работают процессы, как они выполняются, познакомился с несколькими командами для процессов. Также я научился работать с open, read, write.