Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №2 по курсу**

**«Операционные системы»**

**Межпроцессорное взаимодействие**

Студент: Гаврилов Максим Сергеевич

Группа: М8О–206Б–20

Вариант: 21

Преподаватель: Соколов Андрей Алексеевич

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

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 решения 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 файлов lab2\_try.c, child\_proc\_one.c. Используется заголовочные файлы: stdio.h, unistd.h, stdlib.h, string.h, fcntl.h. В программе используются следующие системные вызовы:

1. **read –** пытается записать *n* байт из файлового дескриптора *fd* в буфер *buf*.
2. **Write –** пытается записать *n* байт из буфера *buf в ф*айловый дескриптор *fd*.
3. **Open –** преобразует путь к файлу в файловый дескриптор.
4. **Close –** закрывает файловый дескриптор, который после этого не ссылается ни на один и файл и может быть использован повторно.
5. **Pipe –** создает пару файловых дескрипторов, указывающих на запись именованного канала, и помещает их в массив, на который указывает *filedes*. *filedes[0]* предназначен для чтения, а *filedes[1]* предназначен для записи
6. **Fork –** порождает новый процесс (процесс-потомок), который почти идентичен порождающему процессу-родителю.

**Общий метод и алгоритм решения**.

Для реализации поставленной задачи необходимо:

1. Освоить работу с **fork** и **pipe**
2. Освоить работу с семейством функций **exec**
3. научиться производить ввод-вывод с помощью **write** и **read**

**Основные файлы программы**

**Lab2\_try.c**

|  |
| --- |
| //Гаврилов М.С. М8О-206Б-19 | Лр#2, вар. 21  // Родительский процесс создает два дочерних процесса. Первой строкой пользователь  // в консоль родительского процесса вводит имя файла, которое будет использовано для  // открытия File с таким именем на запись для child1. Аналогично для второй строки  // и процесса child2. Родительский и дочерний процесс должны быть представлены разными  // программами.  // Родительский процесс принимает от пользователя строки произвольной длины и пересылает  // их в pipe1 или в pipe2 в зависимости от правила фильтрации. Процесс child1 и  // child2 производят работу над строками. Процессы пишут результаты своей работы в  // стандартный вывод.  // Правило фильтрации: нечетные строки отправляются в pipe1, четные в pipe2. Дочерние  // процессы инвертируют строки.  #include <string.h>  #include "unistd.h"  #include "stdio.h"  #include "stdlib.h"  #include "fcntl.h"  static int swtcr1 **=** 0**;**  static int swtcr2 **=** 0**;**  int main**()**  **{**  write**(**1**,**"Input two filenames\nthen input strings to be inverted\n"**,**strlen**(**"Input two filenames\nthen input strings to be inverted\n"**));**  char filename1**[**100**];**  char filename2**[**100**];**  //  char buf **=** 0**;** int i **=** 0**;**  **while(**buf **!=** '\n'**)**  **{**  read**(**0**,&**buf**,**1**);**  filename1**[**i**]** **=** buf**;**  **++**i**;**  **}**  filename1**[**i**-**1**]** **=** 0**;**  **for(;;)**  **{**  char buf **=** 0**;** i **=** 0**;**  **while(**buf **!=** '\n'**)**  **{**  read**(**0**,&**buf**,**1**);**  filename2**[**i**]** **=** buf**;**  **++**i**;**  **}**  filename2**[**i**-**1**]** **=** 0**;**    **if(**strcmp**(**filename1**,**filename2**)==**0**)**  **{**  write**(**1**,**"please input string differ to name one\n"**,**strlen**(**"please input string differ to name one\n"**));**  **}**  **else**  **{**  **break;**  **}**  **}**  int l **=** strlen**(**filename1**);**  **for(**int j**=**0**;**j**<**l**;++**j**)**  **{**  **if(**filename1**[**j**]==**'\n'**)**  **{**  filename1**[**j**]** **=** 0**;**  **}**  **}**  l **=** strlen**(**filename2**);**  **for(**int j**=**0**;**j**<**l**;++**j**)**  **{**  **if(**filename2**[**j**]==**'\n'**)**  **{**  filename2**[**j**]** **=** 0**;**  **}**  **}**  int a **=** open**(**filename1**,**O\_CREAT **|** O\_WRONLY **|** O\_TRUNC**,** 00660**);**  write**(**a**,**"This is file one\n(even)\n\n"**,**strlen**(**"This is file one\n(even)\n\n"**));**  close**(**a**);**  a **=** open**(**filename2**,**O\_CREAT **|** O\_WRONLY **|** O\_TRUNC**,** 00660**);**  write**(**a**,**"This is file two\n(odd)\n\n"**,**strlen**(**"This is file two\n(odd)\n\n"**));**  close**(**a**);**  int fd1**[**2**];**  pipe**(**fd1**);** //pipe привязывает к своим концам элементы fd ([0] - выход, [1] - вход)  write**(**1**,**"initiating process separation\n"**,**strlen**(**"initiating process separation\n"**));**  int id **=** fork**();** //разделение процесса, в основном процессе id будет соответствовать pid порожденного, в порожденном id = 0  **if(**id **==** **-**1**)** //ошибка  **{**  write**(**1**,**"error\n"**,**strlen**(**"error\n"**));**  **return** **-**1**;**  **}**  **else**  **if(**id **==** 0**)**  **{**  char arg1**[**1**];** arg1**[**0**]** **=** fd1**[**0**];**  execl**(**"chid\_proc\_one.out"**,**filename1**,**arg1**,NULL);**  **}**  **else**  **{**  //код основного процесса  int fd2**[**2**];**  pipe**(**fd2**);**  int id2 **=** fork**();**  **if(**id2 **==** **-**1**)**  **{**  write**(**1**,**"error in genetating str 2\n"**,**strlen**(**"error in genetating str 2\n"**));**  **return** **-**1**;**  **}**  **else**  **if(**id2 **==** 0**)**  **{**  char arg2**[**1**];** arg2**[**0**]** **=** fd2**[**0**];**  execl**(**"chid\_proc\_two.out"**,**filename2**,**arg2**,NULL);**  **}**  **else**  **{**  //основной процесс  write**(**1**,**"parent: It's parent\n"**,** strlen**(**"parent: It's parent\n"**));**  char inp**[**100**];**  int x**,**y**;**  int stp **=** 0**;**  **while(**stp**==**0**)**  **{**  char buf **=** 0**;** i **=** 0**;**  **while(**buf **!=** '\n'**)**  **{**  **if(**read**(**0**,&**buf**,**1**)==**0**)**  **{**  stp **=** 1**;**  **break;**  **}**  inp**[**i**]** **=** buf**;**  **++**i**;**  **}**  inp**[**i**-**1**]** **=** 0**;**  int l **=** strlen**(**inp**);**  **for(**int j**=**0**;**j**<**l**;++**j**)**  **{**  **if(**inp**[**j**]==**'\n'**)**  **{**  inp**[**j**]** **=** 0**;**  **}**  **}**  l **=** strlen**(**inp**);**  **if(**l **%** 2 **==** 0**)**  **{**  write**(**fd1**[**1**],**inp**,sizeof(**char**)\*(**l**+**1**));**  **}**  **else**  **{**  write**(**fd2**[**1**],**inp**,sizeof(**char**)\*(**l**+**1**));**  **}**  strcpy**(**inp**,**""**);**  **}**  char stop**[**1**];**  stop**[**0**]** **=** 1**;**  write**(**fd1**[**1**],**stop**,sizeof(**char**));**  write**(**fd2**[**1**],**stop**,sizeof(**char**));**  close**(**fd1**[**0**]);**  close**(**fd1**[**1**]);**  close**(**fd2**[**0**]);**  close**(**fd2**[**1**]);**  write**(**1**,**"parent: stop\n"**,**strlen**(**"parent: stop\n"**));**  **return** 0**;**  **}**  **}**  //общий код  **return** 0**;**  **}** |

**Child\_proc\_one.c**

|  |
| --- |
| #include "string.h"  #include "unistd.h"  #include "stdio.h"  #include "stdlib.h"  #include "fcntl.h"  int main**(**int argc**,**char**\*** argv**[])**  **{**  int fd1**[**1**];**fd1**[**0**]** **=** argv**[**1**][**0**];**  char filename1**[**100**];** strcpy**(**filename1**,**argv**[**0**]);**  int f1**;**  //код дочернего процесса  write**(**1**,**"It's child one\n"**,**strlen**(**"It's child one\n"**));**  char str**[**100**];**  int i**=**0**;**  **for(;;)**  **{**  f1 **=** open**(**filename1**,**O\_CREAT **|** O\_WRONLY **|** O\_APPEND**,** 00660**);**  // fflush(stdout);  read**(**fd1**[**0**],&**str**[**i**],sizeof(**char**));**  //printf("[%d] read, got %d\n",getpid(),str[i]);  char stop**[**1**];**stop**[**0**]** **=** 1**;**  **if(**str**[**i**]==**stop**[**0**])**  **{**  close**(**fd1**[**0**]);**  close**(**fd1**[**1**]);**  write**(**1**,**"one: stop\n"**,**strlen**(**"one: stop\n"**));**  exit**(**0**);**  **}**  **if(**str**[**i**]==**0**)**  **{**  //работа со строкой  write**(**1**,**"one: string got\n"**,**strlen**(**"one: string got\n\n"**));**  char res**[**strlen**(**str**)+**1**];**  int k **=** 0**;**  **for(**int j **=** strlen**(**str**)-**1**;**j**>=**0**;--**j**,++**k**)**  **{**  res**[**k**]** **=** str**[**j**];**  **}**  res**[**k**]** **=** 0**;**  write**(**1**,**"one: result written\n"**,**strlen**(**"one: result written\n"**));**  write**(**f1**,**res**,**strlen**(**res**));**  write**(**f1**,**"\n"**,**strlen**(**"\n"**));**    i **=** 0**;**  **}**  **else**  **{**  **++**i**;**  **}**  close**(**f1**);**  **}**  close**(**fd1**[**0**]);**  close**(**fd1**[**1**]);**  **}** |

**Пример работы**

**max@max-Swift:~/Рабочий стол/OS/lab2\_try1$ make**

*gcc --std=c99 lab2\_try.c -o exetry.out*

*gcc --std=c99 chid\_proc\_one.c -o chid\_proc\_one.out*

*gcc --std=c99 chid\_proc\_two.c -o chid\_proc\_two.out*

**max@max-Swift:~/Рабочий стол/OS/lab2\_try1$ ./exetry.out**

*Input two filenames*

*then input strings to be inverted*

file\_1

file\_2

*initiating process separation*

*parent: It's parent*

*It's child one*

*It's child two*

string\_one

*one: string got*

*one: result written*

longer\_string

*two: string is got*

*two: result written*

qwerty

*one: string got*

*one: result written*

qwert

*two: string is got*

*two: result written*

qwer

*one: string got*

*one: result written*

qwe

*two: string is got*

*two: result written*

qw

*one: string got*

*one: result written*

q

*two: string is got*

*two: result written*

*one: stop*

*parent: stop*

*two: stop*

**max@max-Swift:~/Рабочий стол/OS/lab2\_try1$ cat file\_1**

*This is file one*

*(even)*

*eno\_gnirts*

*ytrewq*

*rewq*

*wq*

**max@max-Swift:~/Рабочий стол/OS/lab2\_try1$ cat file\_2**

*This is file two*

*(odd)*

*gnirts\_regnol*

*trewq*

*ewq*

*q*

**max@max-Swift:~/Рабочий стол/OS/lab2\_try1$**

**Вывод**

В этой лабораторной работе не так уж много чего-то, о чем можно написать в выводе, оставив его при этом достаточно содержательным. Мне показалось, что это несложная работа, предназначенная в первую очередь для того чтобы учащийся мог познакомиться с работой с процессами на самом базовом уровне. Так как вообще все взаимодействие здесь максимально низкоуровневое, я также смог чуть лучше понять, как вообще устроены некоторые функции в си. Также при выполнении этой работы я впервые применил утилиту make по собственному желанию.