**Московский авиационный институт**

**(национальный исследовательский университет)**

Институт №8 «Информационные технологии и прикладная математика» Кафедра 806 «Вычислительная математика и программирование» Дисциплина «Операционные системы»

**Лабораторная работа №4 по курсу**

**«Операционные системы»**

**Освоение принципов работы с файловыми системами. Обеспечение обмена данных между процессами посредством технологии «File Mapping»**

## Студент: Кондратьев Егор Алексеевич

## Группа: М80-206Б-19

Преподаватель: Миронов Е.С.

Дата:

## Оценка:

Москва, 2021

**Постановка задачи**

**Цель работы:**

Приобретение практических навыков в:

* Освоение принципов работы с файловыми системами
* Обеспечение обмена данных между процессами посредством технологии «File mapping»

**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 через отображаемые файлы (memory-mapped files). Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

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.

Правило фильтрации: нечетные строки отправляются child1, четные child2. Дочерние процессы удаляют все гласные из строк.

**Общие сведения о программе**

Программа компилируется из файла main.c. Отличительной особенностью программы (относительно кода лабораторной №2) является использование системного вызова mmap. Этот системный вызов запускается с ключом MAP\_ANONYMOUS, это означает, что файл, отображаемый в память, будет виртуальным. Также в этой лабораторной используется обработчик системного сигнала (сигнала о завершении работы родительского процесса).

**Общий метод и алгоритм решения**

Для реализации поставленной задачи необходимо:

1. Изучить принципы работы mmap.
2. Изучить принципы работы системных сигналов.
3. Переписать функции, заменив pipe-ы на mmap.
4. Написать обработчики сигналов.
5. Написать обработку ошибок
6. Написать тесты
7. **Основные файлы программы**

#include <fcntl.h>

#include <signal.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/mman.h>

#include <sys/wait.h>

#include <unistd.h>

int handler = 0;

const int OFFSET = sizeof(int);

void update\_handler(int num){

handler = 1;

}

void child\_work(char \*from, int to){

int idx = OFFSET;

while (1){

if (idx < OFFSET + ((int \*)from)[0]){

char c = from[idx];

if (c != 'a' && c != 'e' && c != 'i' && c != 'o' && c != 'u' && c != 'y' &&

c != 'A' && c != 'E' && c != 'I' && c != 'O' && c != 'U' && c != 'Y'){

write(to, &c, 1);

}

++idx;

}

else{

if (handler){

break;

}

}

}

close(to);

}

void parrent\_work(pid\_t child1, char \*child\_map1, pid\_t child2, char \*child\_map2){

char ch;

int is\_even = 0;

int idx1 = OFFSET;

int idx2 = OFFSET;

while (read(STDIN\_FILENO, &ch, 1) > 0){

if (!is\_even){

child\_map1[idx1++] = ch;

}

else{

child\_map2[idx2++] = ch;

}

if (ch == '\n'){

if (!is\_even){

((int \*)child\_map1)[0] = idx1 - OFFSET;

}

else{

((int \*)child\_map2)[0] = idx2 - OFFSET;

}

is\_even = !is\_even;

}

}

kill(child1, SIGUSR1);

kill(child2, SIGUSR1);

int res1;

int res2;

waitpid(child1, &res1, 0);

waitpid(child2, &res2, 0);

if (res1 != 0 || res2 != 0){

fprintf(stderr, "Something went wrong!\n%d %d\n", res1, res2);

}

}

int open\_file(){

const size\_t NAME\_SIZE = 64;

char f\_name[NAME\_SIZE];

char buf[1];

int idx = 0;

while (idx < NAME\_SIZE && read(STDIN\_FILENO, buf, 1) > 0){

if (buf[0] == '\n'){

break;

}

f\_name[idx++] = buf[0];

}

f\_name[idx++] = '\0';

return open(f\_name, O\_WRONLY | O\_TRUNC | O\_CREAT);

}

void \*check\_map\_creation(){

void \*m\_file = mmap(NULL, 2048, PROT\_READ | PROT\_WRITE, MAP\_SHARED | MAP\_ANONYMOUS, -1, 0);

if (m\_file == MAP\_FAILED){

perror("Cannot create mmap");

exit(-2);

}

((int \*)m\_file)[0] = 0;

return m\_file;

}

void add\_signals(){

void (\*func)(int);

func = signal(SIGUSR1, update\_handler);

if (func == SIG\_IGN){

perror("Cannot add signal");

exit(-4);

}

}

int main(int argc, char \*argv[]){

add\_signals();

int f1 = open\_file();

if (f1 == -1){

perror("File not open");

exit(-1);

}

int f2 = open\_file();

if (f2 == -1){

perror("File not open");

exit(-2);

}

char \*m\_file1 = check\_map\_creation();

pid\_t child1 = fork();

if (child1 == -1){

perror("Cannot create process");

exit(-3);

}

if (child1 == 0){

close(f2);

child\_work(m\_file1, f1);

return 0;

}

close(f1);

char \*m\_file2 = check\_map\_creation();

pid\_t child2 = fork();

if (child2 == -1){

perror("Cannot create process");

exit(-4);

}

if (child2 == 0){

child\_work(m\_file2, f2);

return 0;

}

close(f2);

parrent\_work(child1, m\_file1, child2, m\_file2);

return 0;

}

**Пример работы**

***Первый тест - проверка на обработку случая отсутствия первого файла***

root@Du:/mnt/c/Users/egork/Desktop/os/lab\_2/src$ cat test1

f

root@Du:/mnt/c/Users/egork/Desktop/os/lab\_2/src$ ./a.out <test1

File not found

***Второй тест - проверка на обработку случая отсутствия второго файла***

root@Du:/mnt/c/Users/egork/Desktop/os/lab\_2/src$ cat test2

f1

f

root@Du:/mnt/c/Users/egork/Desktop/os/lab\_2/src$ ./a.out <test2

File not found

***Третий и четвертый тесты - проверка работоспособности программы на корректных данных***

root@Du:/mnt/c/Users/egork/Desktop/os/lab\_2/src$ cat test3

f1

f2

a b c d e f g h i j k l

m n o p q r s t u v w x y z

root@Du:/mnt/c/Users/egork/Desktop/os/lab\_2/src$ ./a.out <test3

root@Du:/mnt/c/Users/egork/Desktop/os/lab\_2/src$ cat f1

b c d f g h j k l

root@Du:/mnt/c/Users/egork/Desktop/os/lab\_2/src$ cat f2

m n p q r s t v w x z

root@Du:/mnt/c/Users/egork/Desktop/os/lab\_2/src$ cat test4

f1

f2

m n o p q r s t u v w x y z

a b c d e f g h i j k l

m n o p q r s t u v w x y z

a b c d e f g h i j k l

m n o p q r s t u v w x y z

a b c d e f g h i j k l

m n o p q r s t u v w x y z

a b c d e f g h i j k l

root@Du:/mnt/c/Users/egork/Desktop/os/lab\_2/src$ ./a.out <test4

root@Du:/mnt/c/Users/egork/Desktop/os/lab\_2/src$ cat f1

m n p q r s t v w x z

m n p q r s t v w x z

m n p q r s t v w x z

m n p q r s t v w x z

root@Du:/mnt/c/Users/egork/Desktop/os/lab\_2/src$ cat f2

b c d f g h j k l

b c d f g h j k l

b c d f g h j k l

b c d f g h j k l

***Пятый тест - проверка работоспособности программы на пустых данных***

root@Du:/mnt/c/Users/egork/Desktop/os/lab\_2/src$ cat test5

f1

f2

root@Du:/mnt/c/Users/egork/Desktop/os/lab\_2/src$ ./a.out <test5

root@Du:/mnt/c/Users/egork/Desktop/os/lab\_2/src$ cat f1

root@Du:/mnt/c/Users/egork/Desktop/os/lab\_2/src$ cat f2

1. **Выводы**

Выполнив данную лабораторную работу я научился работать с файлами, отображаемыми в память, и системными сигналами в Си. Файлы, отображаемые в память, являются крайне полезным инструментом, так как в ряде случаев это значительно повышает эффективность работы программы. Системные сигналы оказались полезными при работе над последующими лабораторными.