Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Саратовский государственный технический университет

имени Гагарина Ю.А.»

Институт электронной техники и приборостроения

Кафедра Информационная безопасность автоматизированных систем

Специальность 10.05.03 Информационная безопасность автоматизированных систем

Практическая работа № 3 по дисциплине «Операционная система UNIX»

Тема «Процессы и сигналы OC UNIX»

|  |  |
| --- | --- |
|  | Выполнил: студент 3 курса  учебной группы с1-ИБС 32  очной формы обучения  Солодилов В.В.  Проверила: профессор  кафедры ИБС  Пластун И. Л. |

Саратов 2022

**ЗАДАНИЕ**

18. Процесс 1 открывает существующий файл и порождает потомка 2. Процесс 1 считывает N байт из файла, выводит их на экран и посылает сигнал SIG1 процессу 2. Процесс 2 также считывает N байт из файла, выводит их на экран и посылает сигнал SIG1 процессу 1. Если одному из процессов встречается конец файла, то он посылает другому процессу сигнал SIG2 и они оба завершаются.

**ЛИСТИНГ ПРОГРАММЫ**

#include <stdio.h>

#include <unistd.h>

#include <signal.h>

#include <string.h>

#include <stdlib.h>

int N; //количество считываемых байт

int pid1, pid2; //процессы symbol; //считываемый байт

int symbol; // ,байт

char \* readFile(FILE \*, int, int);//чтение файла

void \* functionProcess1(FILE \*, int);//1 процесс

void \* functionProcess2(FILE \*, int);//2 процесс

int main() {

printf("Enter number of bytes: ");

scanf("%d", &N);

pid1 = getpid();//создаём первый процесс

FILE \*file = fopen("/home/user/Lab3/read.txt", "rb");//открываем файл на чтение

if (file == NULL) { //проверяем открытие файла

printf("File isn't open. Return...");

return -1;

} else {

printf("File is open...\n");

pid2 = fork();//создаем дочерний процесс

if (pid2 == 0) {

pid2 = getpid(); //получаем id дочерного процесса

kill(SIGUSR1, pid1); //завершаем процесс 1

} else {

signal(SIGUSR1, functionProcess1(file, N)); //определяем реакцию для процесса 1

}

}

}

void \* functionProcess1(FILE \*file, int n) {

if(strcmp(readFile(file, n, 1), "EOF") == 0) { //считываем N байт и проверяем окончание файла

fclose(file); //закрываем файл

exit(0);

} else {

signal(SIGUSR1, functionProcess2(file, n)); //передаем сигнал второму процессу

}

return NULL;

}

void \* functionProcess2(FILE \*file, int n) {

if(strcmp(readFile(file, n, 2), "EOF") == 0) { //считываем N байт и проверяем окончание файла

fclose(file); //закрываем файл

exit(0);

} else {

signal(SIGUSR1, functionProcess1(file, n)); //передаем сигнал первому процессу

}

return NULL;

}

char \* readFile(FILE \*file, int n, int id) {

if (id == 1) { //выводим информацию о работающем процессе

printf("\nProcess %d (%d) read %d bytes: ", id, pid1, n);

} else {

printf("\nProcess %d (%d) read %d bytes: ", id, pid2, n);

}

for (int i = 0; i < n; i++) {

symbol = getc(file);

if (symbol == EOF) {

if (feof(file) != 0) { //проверяем конец файла

printf("\n\nFile's reading completed...\n");

return "EOF"; //возвращаем, если конец файла

} else {

printf("\nError of file's reading!\n");

}

} else printf("%c", symbol); //печатаем символ

}

return "EOR"; //возвращаем, если конец считывания

}

**РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

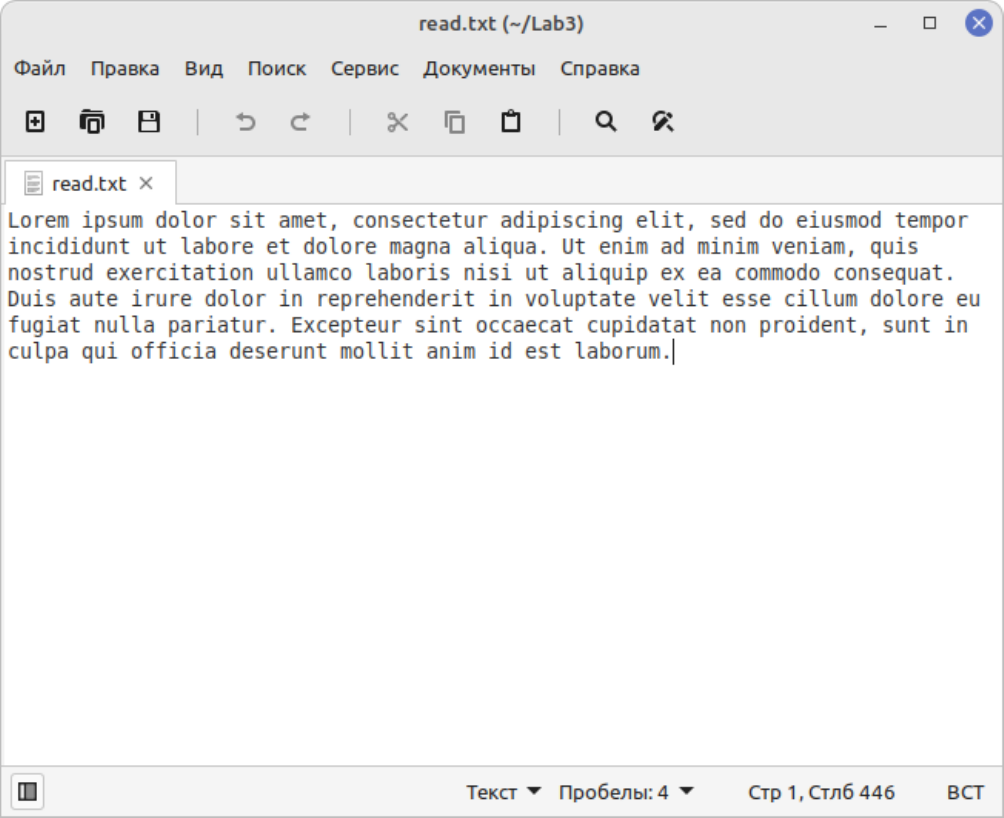


Рисунок 1 – Исходный текст

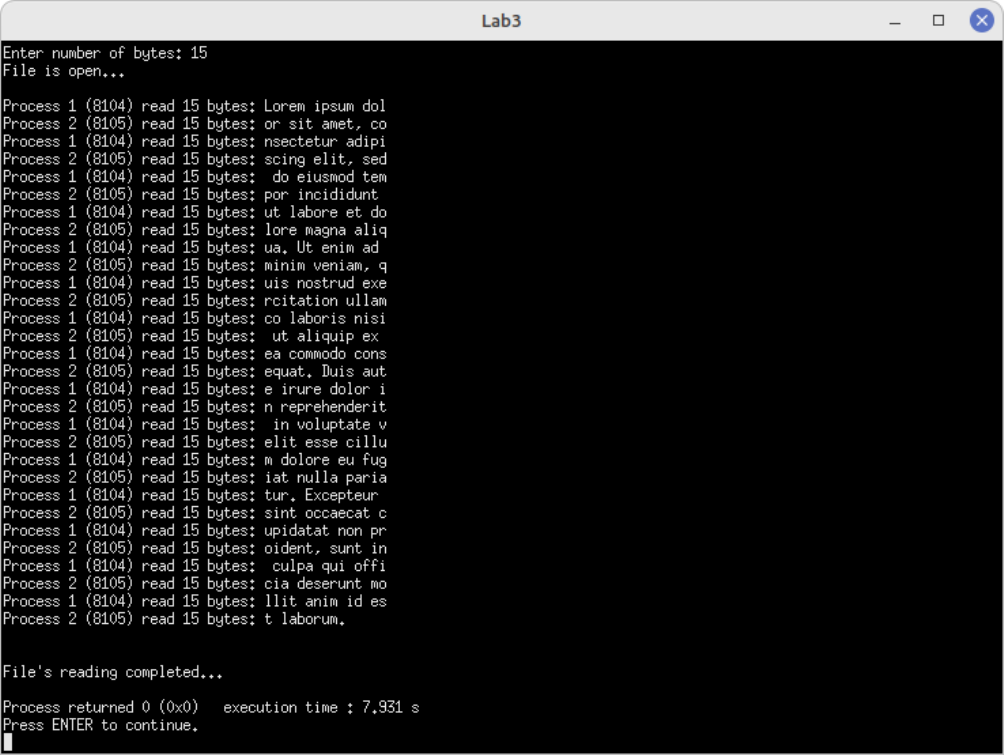


Рисунок 2 – Результат выполнения программы