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»

|  |  |
| --- | --- |
|  | Выполнила: 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 = 20; //число считываемых байт

int pid1, pid2; //процессы

int symbol; //считываемый байт

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

void \* funcProcess1(FILE \*, int);//1 процесс

void \* funcProcess2(FILE \*, int);//2 процесс

void main(void) {

pid1 = getpid();//создаём первый процесс

FILE \*file1 = fopen("read.txt", "r");//открываем файл на чтение

if (file1 == NULL) { //проверяем открытие файла

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

return;

} else {

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

pid2 = fork();//создаем дочерний процесс

if (pid2 == 0) { //процесс 2

pid2 = getpid();

sleep(1);

signal(SIGUSR1, funcProcess2(file1, N)); //определяем реакцию для процесса 2

} else { //процесс 1

signal(SIGUSR1, funcProcess1(file1, N)); //определяем реакцию для процесса 1

}

}

}

void \* funcProcess1(FILE \*file1, int n) {

if(strcmp(readFile(file1, n, 1), "EOF") == 0) { //проверяем окончание файла

fclose(file1); //закрываем файл

signal(SIGUSR2, pid2); //посылаем сигнал процессу 2

printf("\nProcess 1 (%d) is completed with code %d", pid1, signal(SIGTERM, pid1)); //завршаем 1 процесс

printf("\nProcess 2 (%d) is completed with code %d\n", pid2, kill(pid2, SIGCHLD)); //завршаем 2 процесс

exit(0);

} else {

signal(SIGUSR1, funcProcess2(file1, n)); //передаем сигнал второму процессу

}

return NULL;

}

void \* funcProcess2(FILE \*file1, int n) {

if(strcmp(readFile(file1, n, 2), "EOF") == 0) { //считываем N байт и проверяем окончание файла

fclose(file1); //закрываем файл

signal(SIGUSR2, pid1); //посылаем сигнал процессу 1

printf("\nProcess 2 (%d) is completed with code %d", pid2, kill(pid2, SIGCHLD)); //завршаем 2 процесс

printf("\nProcess 1 (%d) is completed with code %d\n", pid1, signal(SIGTERM, pid1)); //завршаем 1 процесс

exit(0);

} else {

signal(SIGUSR1, funcProcess1(file1, n)); //передаем сигнал первому процессу

}

return NULL;

}

char \* readFile(FILE \*file1, 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(file1); //считываем байт из файла

if (symbol == EOF) {

if (feof(file1) != 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");

}

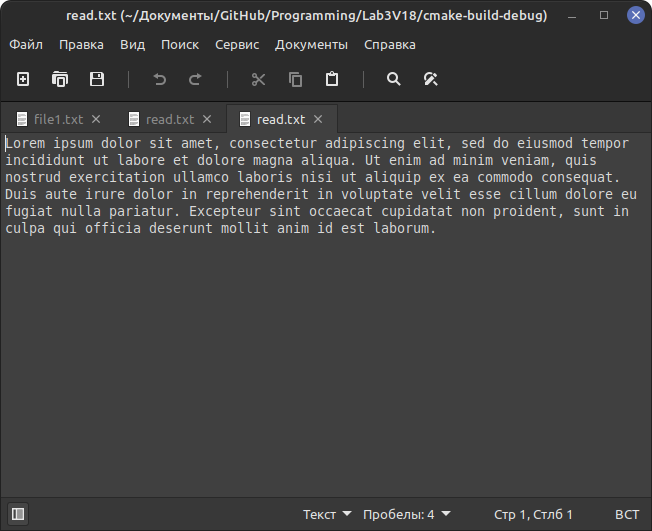
} else printf("%d ", symbol); //печатаем символ

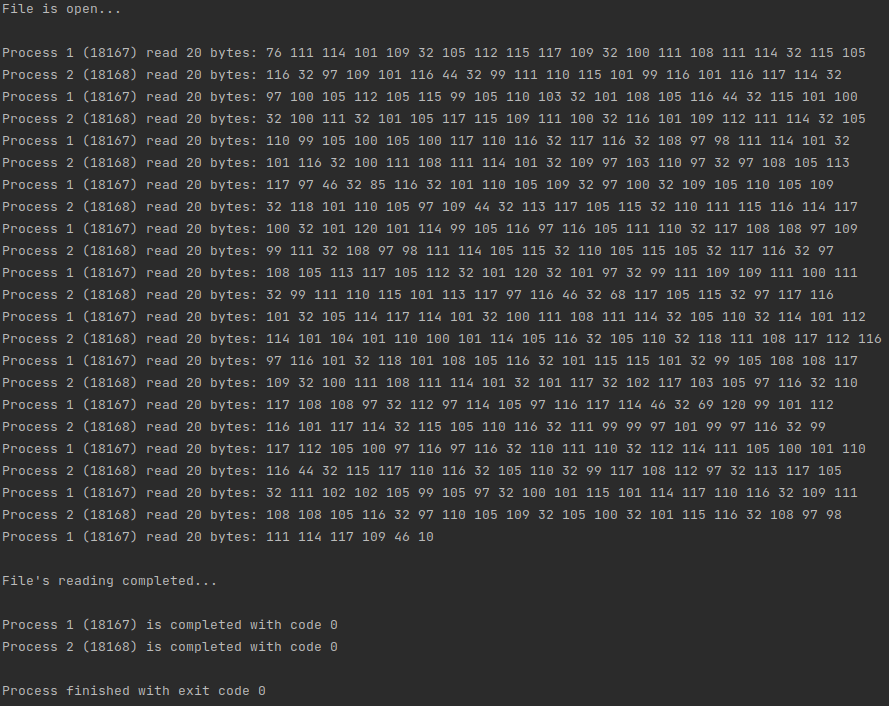
}

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

}

**ЛИСТИНГ ПРОГРАММЫ**

Рисунок 1 – Исходный текст

Рисунок 2 – Результат выполнения программы