**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ» (МАИ)**

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра 304 «Вычислительные машины, системы и сети»

**Лабораторная работа №4**

по дисциплине

**«Операционные системы»**

На тему:

**«Межпроцессорное взаимодействие»**

Выполнили студенты группы М30-325Бк-18

Назарчук Г. В.,

Черноморец М. С.

Проверил:

доцент кафедры 304, к.т.н. Охотников Д. А.

Москва 2021

**Задание**

1. Написать следующие программы:
   1. Программа 1. Программа читает заданный файл и выводит его содержимое в поток вывода.
   2. Программа 2. Программа запускает два экземпляра программы 1 и разными исходными файлами осуществляет с выходными данными побитовую операцию XOR и сохраняет результат в файл.
2. Запустить вторую программу и подать на вход текстовый файл и файл, содержащий случайную последовательность.
3. Запустить вторую программу с файлом, полученным во втором пункте и файлом, содержащим случайную последовательность.
4. Сравнить текстовый файл до и после шифрования.

**Описание программы**

Программа 1 принимает на вход имя файла. Если файл оказывается недоступен, она выведет сообщение об ошибке. Если файл удалось открыть для чтения, то программа считывает данные из файла и переводит в поток вывода.

Программа 2 принимает на вход имена трех файлов:

1. файл, который будет подан на шифровку;
2. файл, содержащий шифр;
3. файл, в который будут записаны зашифрованные данные из файла 1.

Если 1 или 2 файла нет или введены имена для меньше чем 3 файлов – программа выдаст ошибку. Иначе, программа 2 с помощью программы 1 считает и сохранит данные из файлов 1 и 2. После чего программа 2 сравнит размеры шифруемого файла и файла-шифратора и если файл 1 больше файла 2, программа предупредит об этом и предложит пользователю прекратить шифрование.  
Если шифрование продолжено, то программа выполнит посимвольную операцию XOR и результат запишет в файл 3.

**Код программы 1**

#include <unistd.h>

#include <fcntl.h>

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

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

{

if (access(argv[1], F\_OK) == -1)

{

printf("FILE ISN'T ACCESSABLE!");

return -1;

}

execlp("cat", "cat", argv[1], (char\*)0);

return 0;

}

**Код программы 2**

#include <unistd.h>

#include <fcntl.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <sys/wait.h>

#include <stdlib.h>

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#define BUFSIZE 1048575

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

{

int file\_data;

int gamma\_data;

int file\_pipes[2];

int gamma\_pipes[2];

char filebuf[BUFSIZE+1];

char gammabuf[BUFSIZE+1];

char tmpbuf[BUFSIZE+1];

pid\_t fork\_result;

memset(filebuf, '\0', sizeof(filebuf));

memset(gammabuf, '\0', sizeof(gammabuf));

memset(tmpbuf, '\0', sizeof(tmpbuf));

if ((argc <= 3) || (access(argv[1], F\_OK) == -1) || (access(argv[2], F\_OK) == -1))

{

printf("SOME FILES ARE NOT ACCESSEBLE!\n");

return -1;

}

if (pipe(file\_pipes) != 0)

{

printf("COULDN'T OPEN PIPE!");

return -2;

}

fork\_result = fork();

if (fork\_result == (pid\_t)-1)

{

printf("FORK FAILED!");

return -3;

}

else if (fork\_result == 0)

{

close(1);

dup(file\_pipes[1]);

close(file\_pipes[0]);

execlp("./rdfiles", "./rdfiles", argv[1], (char\*)0);

exit(EXIT\_FAILURE);

}

else

{

int stat\_val;

wait(&stat\_val);

file\_data = read(file\_pipes[0], filebuf, BUFSIZE);

}

close(file\_pipes[0]);

close(file\_pipes[1]);

if (pipe(gamma\_pipes) != 0)

{

printf("COULDN'T OPEN PIPE!");

return -4;

}

fork\_result = fork();

if (fork\_result == (pid\_t)-1)

{

printf("FORK FAILED!");

return -5;

}

else if (fork\_result == 0)

{

close(1);

dup(gamma\_pipes[1]);

close(gamma\_pipes[0]);

execlp("./rdfiles", "./rdfiles", argv[2], (char\*)0);

exit(EXIT\_FAILURE);

}

else

{

int stat\_val;

wait(&stat\_val);

gamma\_data = read(gamma\_pipes[0], gammabuf, BUFSIZE);

}

close(gamma\_pipes[0]);

close(gamma\_pipes[1]);

if (gamma\_data<file\_data)

{

printf("FILE DATA IS BIGGER THEN GAMMA KEY. CONTINUE GAMMING?\nPRINT \"Y\" TO CONTINUE.\nPRINT ANYTHING ELSE TO QUIT.\n");

scanf("%s", tmpbuf);

if((toupper(tmpbuf[0])!='Y')||(tmpbuf[1]!='\0'))

{

printf("STOP GEMMING! QUIT PROGRAMM!\n");

return 0;

}

}

memset(tmpbuf, '\0', sizeof(tmpbuf));

int gammized\_file = open(argv[3], O\_CREAT | O\_WRONLY | O\_TRUNC, S\_IRUSR| S\_IWUSR);

if(gammized\_file == -1)

{

printf("COULDN'T OPEN FILE FOR WRITING GAMMIZED DATA!\n");

return -10;

}

for(int i = 0;i < file\_data; i++)

{

tmpbuf[i] = filebuf[i] ^ gammabuf[i%gamma\_data];

}

write(gammized\_file, &tmpbuf, file\_data);

close(gammized\_file);

printf("GAMMING COMPLETED SUCCESSFULLY!\n");

return 0;

}

**Использованный makefile**

gmfiles: rdfiles

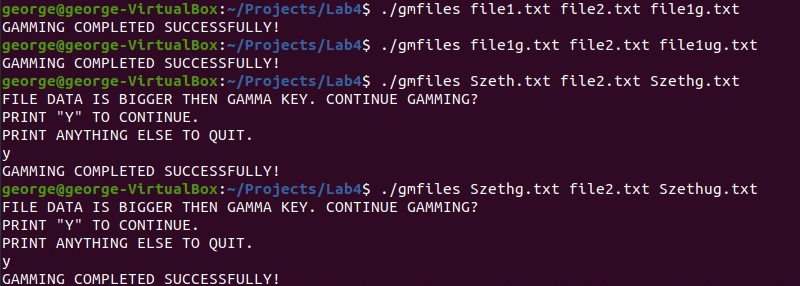
gcc Lab4p2.c -o gmfiles -Wall -fsanitize=address

rdfiles:

gcc Lab4p1.c -o rdfiles -Wall -fsanitize=address

**Пример работы программы**

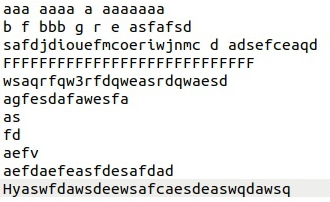
Для зашифровки выбраны два файла: file1.txt и Szeth.txt. file2.txt содержит шифр для шифрования.



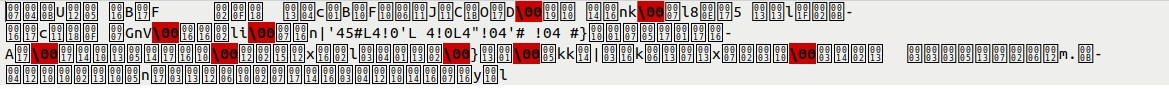
Содержание file2.txt:



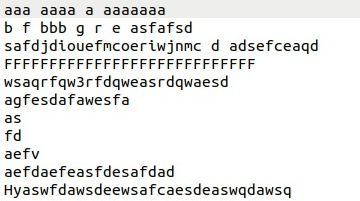
Содержание file1.txt:



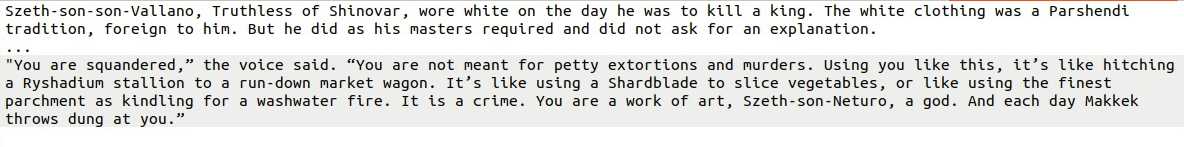
file1.txt после шифрования:



Восстановленный file1.txt:



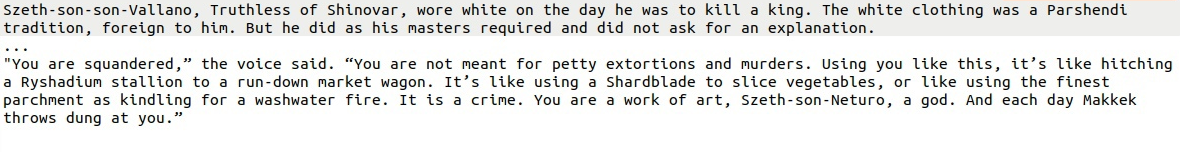
Содержание файла Szeth.txt:



Szeeth.txt после шифрования:



Восстановленный Szeth.txt:



**Список использованной литературы**

1. Основы программирования в Linux: Пер. с англ. / Н. Мэтью, Р. Стоунс. – 4-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2009. – 896 с.: ил.