УО «Белорусский Государственный Технологический Университет»

**ФИТ**

Отчет по Лабораторной работе №4 на тему «Программирование работы с файлами»

Выполнили студенты

8-й группы:

Куницкий Н. Д.

Сакович А. С.

Проверила:

Герман Ю. О

Минск 2016

**Цель**. Освоить системное программирование работы с файлами в Linux.

**Краткие теоретические сведения.**

Прежде всего укажем, как писать программы на языке с и выполнять их в Linux. Используем редактор gedit. Набираем код на языке с в редакторе таким образом. Вызываем редактор

**$ gedit hello.c**

Набираем текст

#include <stdio.h>

main() {

printf(“Hello World\n”);

}

Сохраняем файл в редакторе, например **hello.c**. Будет сохранен в текущем каталоге.

Компилируем файл

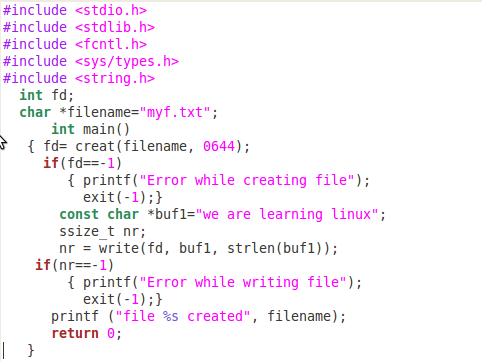
**$ gсс -о hello hello.c**

Запускаем на выполнение

**$ ./hello**



**СОЗДАНИЕ ФАЙЛА**

****

Наберем эту программу и выполним ее. После сохранения файла filepr1.c в окне редактора надо ввести сначала строку для компиляции файла

$gcc –o filepr1 filepr1.c

а затем ввести строку для выполнения файла

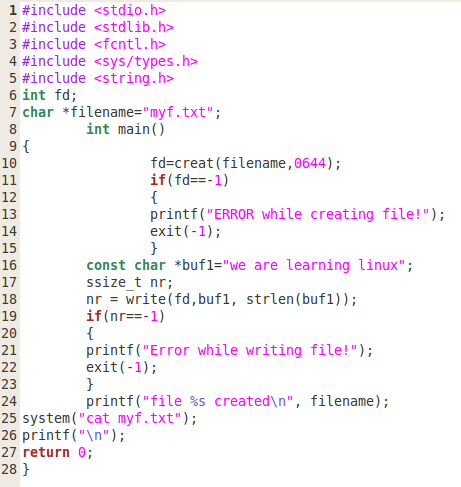
$./filepr1.

Файл создаем с помощью функции операционной системы **creat**. В качестве первого параметра указываем имя файла. Второй параметр представлен числовой константой 644, которая означает, что из файла можем читать, писать и выполнять.

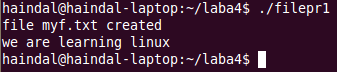
Если файл успешно создан, то в переменной fd указывается целое число, представляющее так называемый дескриптор файла. При возникновении ошибки указывается число -1. Далее мы заносим в буфер строку, которую хотим записать в файл и представленную переменной buf1.

Записываем строку в файл с помощью команды write. Первый аргумент представляет дескриптор файла, второй буфер с подлежащими записи данными, третий – размер буфера.

Теперь несколько изменим нашу программу. Мы добавляем команду **system**(“cat myf.txt”); для выполнения системной функции вывода содержимого файла на консоль.



Скриншот:



Теперь рассмотрим, как читать из файла в массив байтов и выполнять обработку.

**Чтение файла: системный вызов read()**

Системный вызов read(), объявленный в файле unistd.h, позволяет читать данные из файла в отличие от библиотечных функций файлового ввода-вывода, которые предоставляют возможность интерпретации считываемых данных. Можно, например, записать в файл следующее содержимое:

**fscanf (filep, "%s", buffer);**

**fscanf (filep, "%d", number);**

Системный вызов read() читает данные в "сыром" виде, то есть как последовательность байтов, без какой-либо интерпретации. Ниже представлен адаптированный прототип read().

**ssize\_t read (int fd, void \* buffer, size\_t count);**

Первый аргумент - это файловый дескриптор. Здесь больше сказать нечего. Второй аргумент - это указатель на область памяти, куда будут помещаться данные. Третий аргумент - количество байтов, которые функция read() будет **пытаться** прочитать из файла. Возвращаемое значение - количество прочитанных байтов, если чтение состоялось и -1, если произошла ошибка. Заметим, что если read() возвращает значение меньше count, то это не символизирует об ошибке.

Тип size\_t в Linux используется для хранения размеров блоков памяти. Какой тип реально скрывается за size\_t, зависит от архитектуры: как правило, это unsigned long int или unsigned int. Тип ssize\_t (Signed SIZE Type) - это тот же size\_t, только знаковый. Используется, например, в тех случаях, когда нужно сообщить об ошибке, вернув отрицательный размер блока памяти. Системный вызов read() именно так и поступает.

**Задание** **1**. Изменить программу так, чтобы она сначала предложила ввести текст с консоли, а затем записала введенный текст в файл.

#include<stdio.h>

#include<stdlib.h>

#include<fcntl.h>

#include<sys/types.h>

#include<string.h>

int fd;

char filename[20];

int main()

{

printf("Enter name new file (file\_name < 20 symbole):\n");

scanf("%s",filename);

fd=creat(filename,0644);

if(fd==-1)

{

printf("ERROR while creating file!");

exit(-1);

}

char buf1[30];

printf("Enter text to file (text < 30 symbole, width'\_'):\n");

scanf("%s", buf1);

ssize\_t nr;

nr = write(fd,buf1, strlen(buf1));

if(nr==-1)

{

printf("Error while writing file!");

exit(-1);

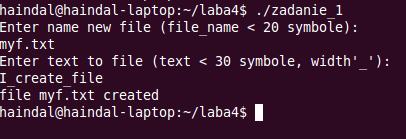
}

printf("file %s created\n", filename);

return 0;

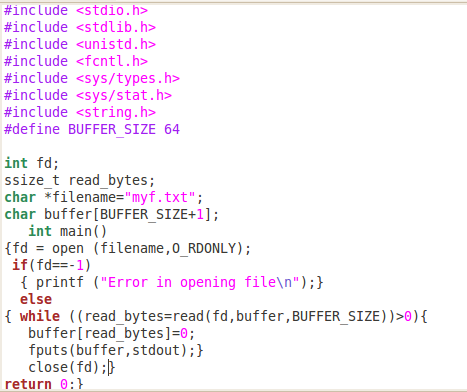
}

Скриншот:



**Задание 2.** Прочитать содержимое файла.

Решение.



Скриншот:



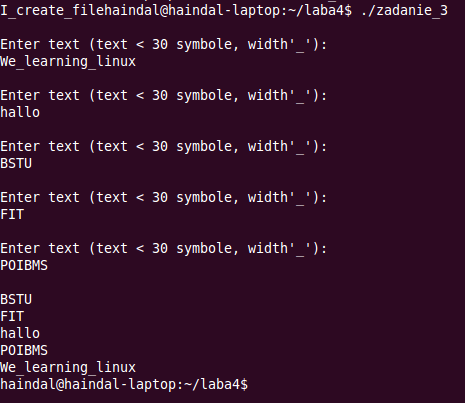
**Задание 3.**

Написать программу, которая считывает содержимое файла и выводит в отсортированном виде.

Листинг:

#include <stdio.h>  
#include <stdlib.h>  
#include <fcntl.h>  
#include <sys/types.h>  
#include <string.h>  
  
int fd;  
char \*filename="file\_zad3.txt";  
ssize\_t nd;  
int main()  
{  
fd=creat(filename, 0644);  
if(fd==-1)  
{  
printf("Error in create file");  
}   
else{  
int i=0;  
while(i<5){  
char buffer[30];  
printf("\nEnter text (text < 30 symbole, width'\_'):\n");  
scanf("%s",buffer);  
char \*x="\n";  
strcat(buffer,x);  
nd=write(fd,buffer,strlen(buffer));  
i++;  
}  
close(fd);  
printf("\n");  
system("sort file\_zad3.txt");  
}  
return 0;  
}

Скриншот:



**Задание 4**. Написать программу, которая выводит меню в виде:

1. Add record in the form Name:Group
2. Find Group by Name
3. Exit

и обрабатывает пункты (пользователь вводит номер пункта).

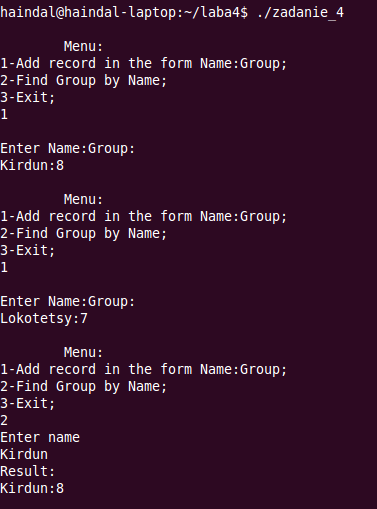
Листинг:

#include <stdio.h>  
#include <stdlib.h>  
#include <fcntl.h>  
#include <sys/types.h>  
#include <string.h>  
  
int fd;  
int counter;  
char \*filename="file\_zad4.txt";  
ssize\_t nd;  
int main()  
{  
fd=creat(filename,O\_RDONLY | O\_WRONLY);  
if(fd==-1)  
{  
printf("Error in creating file");  
}   
else{  
do{  
printf("\n\tMenu:");  
printf("\n1-Add record in the form Name:Group;");  
printf("\n2-Find Group by Name;");  
printf("\n3-Exit;\n");  
scanf("%d",&counter);  
switch(counter)  
{  
case 1:{  
char buffer[30];   
printf("\nEnter Name:Group:\n");  
scanf("%s",buffer);  
char \*x="\n";  
strcat(buffer,x);  
nd=write(fd,buffer,strlen(buffer));  
}break;  
case 2:{  
system("./script\_4");   
}break;  
}  
}while(counter!=3);  
close(fd);}  
return 0;  
}

SCRIPT\_4

#!/bin/bash  
echo "Enter name";  
read name;  
echo "Result:";  
cat file\_zad4.txt | grep $name;

Скриншот



**Задание 5.** В предыдущей программе добавить и запрограммировать пункт «Показать все записи».

Листинг:

#include <stdio.h>  
#include <stdlib.h>  
#include <fcntl.h>  
#include <sys/types.h>  
#include <string.h>  
  
int fd;  
int counter;  
char \*filename="file\_zad4.txt";  
ssize\_t nd;  
int main()  
{  
fd=creat(filename,O\_RDONLY | O\_WRONLY);  
if(fd==-1)  
{  
printf("Error in creating file");  
}   
else{  
do{  
printf("\n\tMenu:");  
printf("\n1-Add record in the form Name:Group;");  
printf("\n2-Find Group by Name;");  
printf("\n3-Show to console;");  
printf("\n4-Exit;\n");  
scanf("%d",&counter);  
switch(counter)  
{  
case 1:{  
char buffer[30];   
printf("\nEnter Name:Group:\n");  
scanf("%s",buffer);  
char \*x="\n";  
strcat(buffer,x);  
nd=write(fd,buffer,strlen(buffer));  
}break;  
case 2:{  
system("./script\_4");   
}break;  
case 3:  
{  
system("./script\_5");  
}break;  
}  
}while(counter!=4);  
close(fd);}  
return 0;  
}

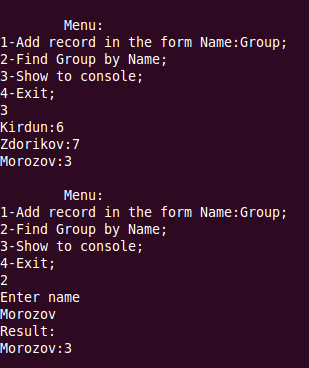
SCRIPT\_5

#!/bin/bash  
cat file\_zad4.txt

SCRIPT\_4

echo "Enter name";  
read name;  
echo "Result:";  
cat file\_zad4.txt | grep $name;

Скриншот:



Вывод:

Мы освоили системное программирование работы с файлами в Linux. Научились создавать файлы, записывать информацию, считывать.