Задание 1.

Проанализировать работу приведенных программ и объяснить результаты их работы.

Программа 1

```
#include <stdio.h>
#include <fcntl.h>

/*
On my machine, a buffer size of 20 bytes
translated into a 12-character buffer.
Apparently 8 bytes were used up by the
stdio library for bookkeeping.
 */

int main()
{
	// have kernel open connection to file alphabet.txt
	int fd = open("alphabet.txt",O_RDONLY);

	// create two a C I/O buffered streams using the above connection
	FILE *fs1 =fdopen(fd,"r");
	charbuff1[20];
	setvbuf(fs1,buff1, IOFBF,20);
	FILE *fs2 =fdopen(fd,"r");
	char buff2[20];
	setvbuf(fs2,buff2,_IOFBF,20);

	// read a char & write it alternatingly from fs1 and fs2
	int flag1 =1, flag2 =2;
	while(flag1 ==1|| flag2 ==1)
	{
		char c;
		flag1 =fscanf(fs1,"%c",&c);
		if(flag1 ==1)
			fprintf(stdout,"%c",c);
		flag2 =fscanf(fs2,"%c",&c);
		if(flag2 ==1)
			fprintf(stdout,"%c",c);
	}
	return0;
}
```

Результат выполнения:

Aubvcwdxeyfzghijklmnopqrst

Описание работы программы:

В результате использования системного вызова open() создается новый файловый дескриптор в системной таблице открытых файлов и запись в таблице открытых файлов процесса. Далее fs1 и fs2 присваивают указатель на структуру FILE для одного файла, то есть ссылаются на одну и ту же запись в системной таблице открытых файлов. Функция setvbuf задает принудительную буферизацию при чтении файла и устанавливает размер буфера равный 20. Вызовы функции fscanf приведут к тому, что при первом вызове из файла будет считана первые 20 символов и значение текущей позиции файла будет смещено на 20. Так как значение текущей позиции файла для fs1 и fs2 одинаковы второй вызов fscanf считает последние 6 символов.Через 7 итераций цикла while выводиться будут только символы из первого буфера, так как второй опустеет. Результатом будет являться строка, где поочередно символы будут печататься то из первого буфера, то из второго.

Программа 2

```
#include <fcntl.h>
#include <unistd.h>
```

```c
int main()
{
	// have kernel open two connection to file alphabet.txt
	int fd1 = open("alphabet.txt",O_RDONLY);
	int fd2 = open("alphabet.txt",O_RDONLY);

	// read a char & write it alternatingly from connections fs1 & fd2
	while(1)
	{
		char c;
		if(read(fd1,&c,1)!=1)break;
			write(STDOUT_FILENO,&c,1);
		if(read(fd2,&c,1)!=1)break;
			write(STDOUT_FILENO,&c,1);
	}
	return0;
}
```

Результатвыполнения:

AAbbccddeeffgghhiijjkkllmmnnooppqqrrssttuuvvwwxxyyzz

Описаниеработыпрограммы:

Системные вызовыopen() создадут 2 разные записи в таблице файлов процесса и 2 разные записи в системой таблицы открытых файлов. У каждой из записей будет своя текущая позиция в файле, поэтому чтение с использованием одной записи не приводит к изменениям значения текущей позиции в другой записи.

Задание2.

Написать программу, которая открывает один и тот же файл два раза с использованием библиотечной функции fopen(). Для этого объявляются два файловых дескриптора. В цикле

записать в файл буквы латинского алфавита поочередно передавая функции fprintf() то первый дескриптор, то – второй.

Программа 3

```
#include <stdio.h>

int main()
{
	FILE* fs[2];
	char c;
	intcurr=0;

	fs[0]=fopen("alphabet.txt","w");
	fs[1]=fopen("alphabet.txt","w");

	for(c ='a'; c <='z';++c)
	{
		fprintf(fs[curr],"%c", c);
		curr=((curr==0)?1:0);
	}

	fclose(fs[1]);
	fclose(fs[0]);
	return0;
}
```

Результат выполнения:

acegikmoqsuwy

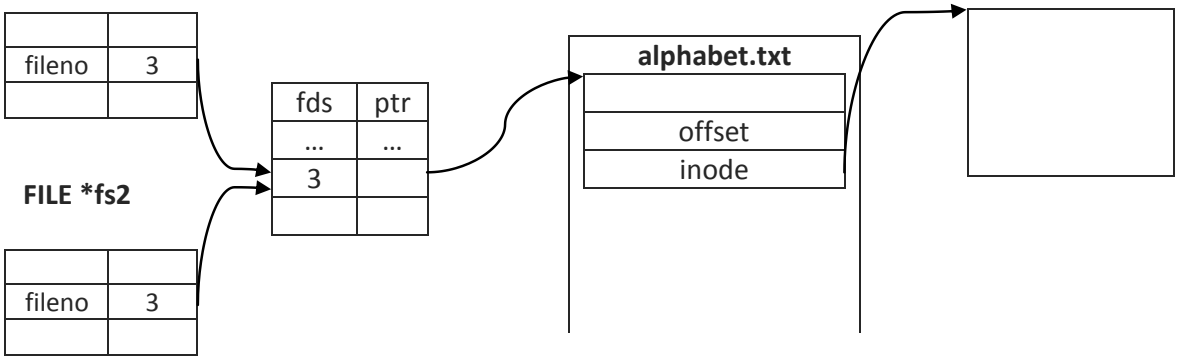
Описание работы программы:

Два вызова функции fopen() с параметром записи "w" создаютдва разных файловых дескриптора к файлу "alphabet.txt", причем если файл существовал, то удаляется связанный с ним inode и создается новый. Для записи в файл используется 2 разных файловых структуры fs[0] и fs[1].В цикле while буфер fs[0] заполняется нечетными символами алфавита, буфер fs[1] – четными. Поскольку используются 2 различных дескриптора, то смещение текущей позиции файла при каждом вызове fprintf() происходит независимо.Так как fprintf() использует буферизацию – окончательная запись в файл осуществляется только при вызове функции fclose() или fflush(), либо при полном заполнении буфера.При вызове fclose(fs[0]) содержимое буфера вывода первой структуры записывается в файл, текущий размер которого изменяется. При этом, поскольку обе структуры были открыты с параметром записи "w" (а не "a", APPEND), во второй структуре значение текущей позиции файла не изменяется. При вызове fclose(fs[1]) из второго буфера вывода символы записываются в файл с позиции начала файла, тем самым затирая информацию, которая была записана до этого из буфера вывода дескриптора fs[0].

Схемы к программам.

1.

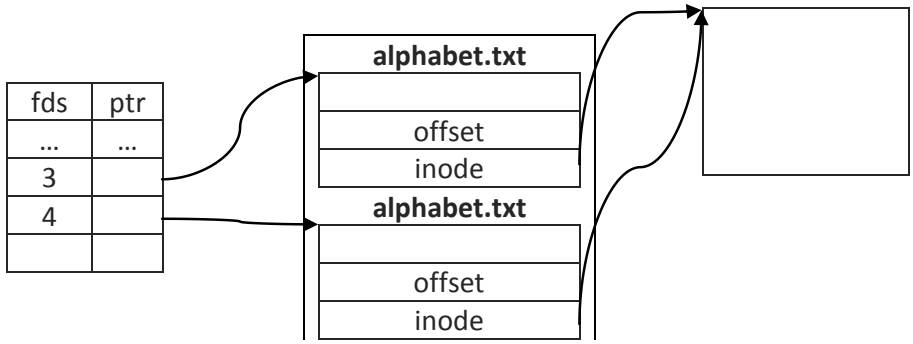
	Таблица открытых файлов процесса	**Системная таблица открытых файлов**	**Диск**
FILE *fs1			**inode**



2.

Таблица открытых файлов процесса **Системная таблица открытых файлов** **Диск**

inode



3.

Таблица открытых файлов процесса **Системная таблица открытых файлов** **Диск**

FILE *fs1

inode